Special Feature 1 →いざ機械学習！
Special Feature 2 →プレゼン力向上
Extra Feature →マストドンやっている？

Software Design

［ソフトウェア デザイン］
OSとネットワーク、
IT環境を支える
エンジニアの総合誌

August 2017

8

2017年8月18日発行
毎月1回18日発行
通巻388号
（発刊322号）
1985年7月1日
第三種郵便物認可
ISSN 0916-6297
定価
本体 1,220円
+税

Extra Feature pixivがPawooを爆速リリースした理由

Mastodon旋風からわかるSNSの未来

構築・運用のノウハウから将来像まで

Special Feature 1

{ディープラーニング
マシンラーニング}

私も機械学習エンジニアになりたい！

先端Web企業の取り組み方は？

Special Feature 2

【保存版】プレゼンテーション技術力の高め方

エンジニアのためのうけるプレゼン・すべるプレゼン

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

目次

Contents

Software Design

August, 2017

8

Special Feature 1 → 第1特集 | Page

私も機械学習エンジニアになりたい!

17

〜先端Web企業の取り組み方は?〜

目次

Contents

Software Design

August, 2017

8

Column

Development

OS/Network

［ロゴデザイン］
デザイン集合ゼブラ+坂井 哲也
［表紙デザイン］
藤井 耕志（Re:D Co.）
［表紙写真］
Eric Isselée / AdobeStock
［イラスト］
＊フクモトミホ
＊高野涼香
［本文デザイン］
＊安達 恵美子
＊石田 昌治（マップス）
＊岩井 栄子
＊ごぽうデザイン事務所
＊近藤 しのぶ
＊SeaGrape
＊轟木 亜紀子、阿保 裕美、
佐藤 みどり、徳田 久美
（トップスタジオデザイン室）
＊伊勢 歩、横山 慎昌（BUCH+）
＊藤井 耕志、萩村 美和（Re:D Co.）
＊森井 一三

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

BOOKS INFOMATION

現場で役立つシステム設計の原則

変更を楽で安全にするオブジェクト指向の実践技法

増田亨 著

A5判／320ページ
定価（本体2940円+税）

ISBN978-4-7741-9087-7

設計次第でソフトウェアの変更作業は楽で安全なものに変わる！
「ソースがごちゃごちゃしていて、どこに何が書いてあるのか理解するまでがたいへん」「1つの修正のために、あっちもこっちも書きなおす必要がある」「ちょっとした変更のはずが、本来はありえない場所にまで影響して、大幅なやり直しになってしまった」といったトラブルが起こるのは、ソフトウェアの設計に問題があるから。日本最大級となる求人情報サイト「イーキャリアJobSearch」の主任設計者であり、システム設計のベテランである著者が、コードの具体例を示しながら、良い設計のやり方と考え方を解説します。

コンテンツ・デザインパターン

吉澤浩一郎 著

B5変形判／208ページ
定価（本体2020円+税）

ISBN978-4-7741-9063-1

SEOにはじまり、SNSやコミュニティにおけるコミュニケーションにおいて必須なものとしてコンテンツの価値に注目が集まっています。しかし、「どうすればコンテンツをつくれるのか？」「どのようなコンテンツをつくればいいのか？」とお悩みの方も多いのではないでしょうか。
本書では、自社商品やサービスを売れるようにするために「誰に・何を・どのように」伝えていくべきかという流れとパターンを体系化し、具体的なコンテンツのつくり方をまとめています。国内外の豊富な事例を収録し、わかりやすく解説。Webマーケティング担当者必携の1冊です。

IBM Bluemix クラウド開発入門

B5変形判／288ページ
定価（本体2800円+税）

ISBN978-4-7741-9084-6

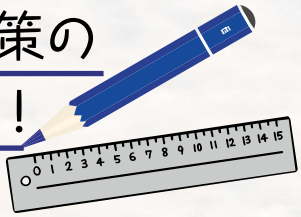
常田秀明・水津幸太・大島騎頼 著
Bluemix User Group 監修

IBMのクラウドサービスであるBluemixを基本的な導入方法から実際のアプリケーションを作る方法まで本書では紹介します。Bluemixの特徴は豊富なサービス群です。データを格納するための各種データストアサービス（RDBMS、NoSQL等）があります。さらにアプリケーションを開発するためのDevOpsサービス群、運用監視を行うための監視機能や、最近話題になっている人工知能利用のためのWatsonAPI等が提供されてます。これらをAPI経由で利用することによりシステムをより効率的に開発できます。

技術評論社 当社書籍・雑誌のご購入は全国の書店、またはオンライン書店でお願い致します。
〒162-0846 東京都新宿区市谷左内町 21-13 販売促進部 TEL.03-3513-6150 FAX.03-3513-6151

あなたを
合格へと導く
一冊があります！

効率よく学習できる
試験対策の
大定番！



岡嶋裕史　著
A5判／624ページ
定価（本体2980円+税）
ISBN978-4-7741-8508-8

金子則彦　著
A5判／688ページ
定価（本体3300円+税）
ISBN978-4-7741-8749-5

岡嶋裕史　著
A5判／680ページ
定価（本体2880円+税）
ISBN978-4-7741-8502-6

エディフィストラーニング株式会社　著
B5判／400ページ
定価（本体2980円+税）
ISBN978-4-7741-9055-6

岡嶋裕史　著
A5判／432ページ
定価（本体1880円+税）
ISBN978-4-7741-8501-9

庄司勝哉・吉川允樹　著
B5判／328ページ
定価（本体1480円+税）
ISBN978-4-7741-9054-9

大滝みや子・岡嶋裕史　著
A5判／744ページ
定価（本体2980円+税）
ISBN978-4-7741-8500-2

加藤昭・高見澤秀幸・矢野龍王　著
B5判／464ページ
定価（本体1780円+税）
ISBN978-4-7741-9053-2

角谷一成・イエローテールコンピュータ　著
A5判／544ページ
定価（本体1680円+税）
ISBN978-4-7741-8495-1

山本三雄　著
B5判／592ページ
定価（本体1480円+税）
ISBN978-4-7741-9052-5

技術評論社　当社書籍・雑誌のご購入は全国の書店、またはオンライン書店でお願い致します。
〒162-0846　東京都新宿区市谷左内町 21-13　販売促進部　TEL.03-3513-6150 FAX.03-3513-6151

紙面版
A4判・16頁
オールカラー

電脳会議

DENNOUKAIGI

一切無料

新規購読会員受付中!

『電脳会議』は情報の宝庫、世の中の動きに遅れるな!

『電脳会議』は、年6回の不定期刊行情報誌です。A4判・16頁オールカラーで、弊社発行の新刊・近刊書籍・雑誌を紹介しています。この『電脳会議』の特徴は、単なる本の紹介だけでなく、著者と編集者が協力し、その本の重点や狙いをわかりやすく説明していることです。平成17年に「通巻100号」を超え、現在200号に迫っている、出版界で評判の情報誌です。

今が旬の情報満載!

新規送付のお申し込みは…

Web検索か当社ホームページをご利用ください。
Google、Yahoo!での検索は、

電脳会議事務局　検索

当社ホームページからの場合は、
https://gihyo.jp/site/inquiry/dennou
と入力してください。

一切無料!

◉「電脳会議」紙面版の送付は送料含め費用は一切無料です。そのため、購読者と電脳会議事務局との間には、権利&義務関係は一切生じませんので、予めご了承下さい。

毎号、厳選ブックガイドも付いてくる!!

『電脳会議』とは別に、1テーマごとにセレクトした優良図書を紹介するブックカタログ（A4判・4頁オールカラー）が2点同封されます。扱われるテーマも、自然科学／ビジネス／起業／モバイル／素材集などなど、弊社書籍を購入する際に役立ちます。

株式会社 技術評論社　電脳会議事務局　〒162-0846　東京都新宿区市谷左内町21-13

OSとネットワーク、
IT環境を支えるエンジニアの総合誌

Software Design

毎月18日発売

PDF電子版
Gihyo Digital
Publishingにて
販売開始

年間定期購読と電子版販売のご案内

1年購読（12回）

14,880円（税込み、送料無料）1冊あたり1,240円（6%割引）

PDF電子版の購入については

Software Design ホームページ
http://gihyo.jp/magazine/SD

をご覧ください。
PDF電子版の年間定期購読も受け付けております。

- インターネットから最新号、バックナンバーも1冊からお求めいただけます！
- 紙版のほかにデジタル版もご購入いただけます！
  デジタル版はPCのほかにiPad/iPhoneにも対応し、購入するとどちらでも追加料金を支払うことなく読むことができます。

※ご利用に際しては、／˜＼Fujisan.co.jp（http://www.fujisan.co.jp/）に記載の利用規約に準じます。

Fujisan.co.jpからのお申し込み方法

1 >> ／˜＼Fujisan.co.jpクイックアクセス
http://www.fujisan.co.jp/sd/

2 >> 定期購読受付専用ダイヤル
0120-223-223（年中無休、24時間対応）

Software Design plus

Software Design plusシリーズは、OSとネットワーク、IT環境を支えるエンジニアの総合誌『Software Design』編集部が自信を持ってお届けする書籍シリーズです。

最新刊！

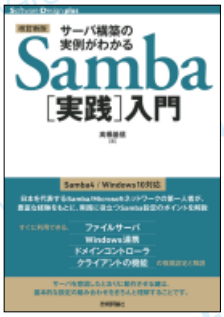
常田秀明、水津幸太、大島騎頼 著、Bluemix User Group 監修
B5変形判・288ページ
定価 2,800円（本体）+税
ISBN 978-4-7741-9084-6

打田智子、大須賀稔、大杉直也、西潟一生、西本順平、平賀一昭 著
B5変形判・392ページ
定価 3,800円（本体）+税
ISBN 978-4-7741-8930-7

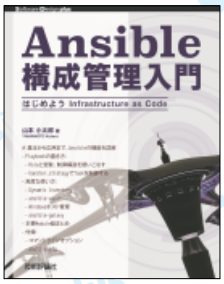
養成読本編集部 編
B5判・144ページ
定価1,780円（本体）+税
ISBN 978-4-7741-8865-2

星野武史 著、花井志生 監修
A5判・256ページ
定価 2,580円（本体）+税
ISBN 978-4-7741-8729-7

小川晃通 著
A5判・272ページ
定価 2,280円（本体）+税
ISBN 978-4-7741-8570-5



髙橋基信 著
A5判・256ページ
定価 2,680円（本体）+税
ISBN 978-4-7741-8000-7



山本小太郎 著
B5変形判・176ページ
定価 2,480円（本体）+税
ISBN 978-4-7741-8885-0

中井悦司 著
B5変形判・272ページ
定価 2,980円（本体）+税
ISBN 978-4-7741-8426-5

前橋和弥 著
B5変形判・304ページ
定価 2,680円（本体）+税
ISBN 978-4-7741-8188-2

五味弘 著
B5変形判・272ページ
定価 2,580円（本体）+税
ISBN 978-4-7741-8035-9

養成読本編集部 編
B5判・168ページ
定価1,980円（本体）+税
ISBN 978-4-7741-8360-2

養成読本編集部 編
B5判・232ページ
定価2,080円（本体）+税
ISBN 978-4-7741-8385-5

養成読本編集部 編
B5判・200ページ
定価2,080円（本体）+税
ISBN 978-4-7741-8034-2

養成読本編集部 編
B5判・112ページ
定価 1,980円（本体）+税
ISBN 978-4-7741-7992-6

Dockerエキスパート養成読本
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-7441-9

サーバ／インフラエンジニア養成読本 基礎スキル編
福田和宏、中村文則、竹本浩、木本裕紀 著
定価 1,980円+税　ISBN 978-4-7741-7345-0

Laravelエキスパート養成読本
川瀬裕久、古川文生、松尾大、竹澤有貴、小山哲志、新原雅司 著
定価 1,980円+税　ISBN 978-4-7741-7313-9

Pythonエンジニア養成読本
鈴木たかのり、清原弘貴、嶋田健志、池内孝啓、関根裕紀、若山史郎 著
定価 1,980円+税　ISBN 978-4-7741-7320-7

データサイエンティスト養成読本 R活用編
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-7057-2

Javaエンジニア養成読本
きしだなおき、のざきひろふみ、吉田真也、菊田洋一、渡辺修司、伊賀敏樹 著
定価 1,980円+税　ISBN 978-4-7741-6931-6

JavaScriptエンジニア養成読本
吾郷協、山田順久、竹馬光太郎、智大二郎 著
定価 1,980円+税　ISBN 978-4-7741-6797-8

WordPress プロフェッショナル養成読本
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6787-9

サーバ／インフラエンジニア養成読本 ログ収集〜可視化編
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6983-5

フロントエンドエンジニア養成読本
養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-6578-3

PHPライブラリ&サンプル実践活用［厳選100］
WINGSプロジェクト 著
定価 2,480円+税　ISBN 978-4-7741-6566-0

Hyper-V仮想化技術活用ガイド
遠山藤乃 著
定価 3,500円+税　ISBN 978-4-7741-6571-4

改訂版　Zabbix統合監視実践入門
寺島広大 著
定価 3,500円+税　ISBN 978-4-7741-6543-1

Vyatta仮想ルータ活用ガイド
松本直人、さくらインターネット研究所、日本Vyattaユーザー会 著
定価 3,300円+税　ISBN 978-4-7741-6553-0

技術評論社

養成読本編集部 編
B5判・112ページ
定価2,180円(本体)+税
ISBN 978-4-7741-8894-2

養成読本編集部 編
B5判・192ページ
定価1,980円(本体)+税
ISBN 978-4-7741-8863-8

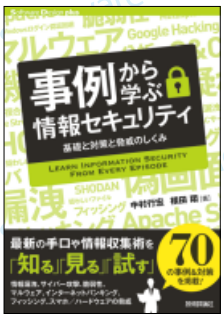
養成読本編集部 編
B5判・160ページ
定価2,180円(本体)+税
ISBN 978-4-7741-8895-9

養成読本編集部 編
B5判・240ページ
定価 1,980円(本体)+税
ISBN 978-4-7741-8877-5

香山哲司、小野寺匠 著
A5判・176ページ
定価 2,480円(本体)+税
ISBN 978-4-7741-8815-7

高宮安仁、鈴木一哉、松井暢之、
村木暢哉、山崎泰宏 著
A5判・352ページ
定価 3,200円(本体)+税
ISBN 978-4-7741-7983-4

斎藤祐一郎 著
A5判・160ページ
定価 2,280円(本体)+税
ISBN 978-4-7741-7865-3

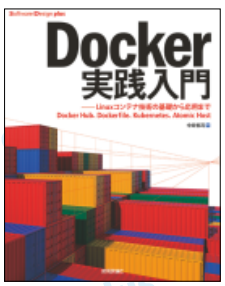


中村行宏、横田翔 著
A5判・320ページ
定価 2,480円(本体)+税
ISBN 978-4-7741-7114-2

川本安武 著
A5判・400ページ
定価 2,980円(本体)+税
ISBN 978-4-7741-6807-4

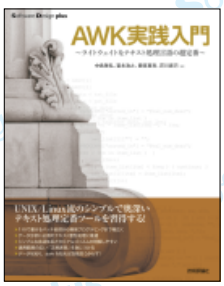
勝俣智成、佐伯昌樹、
原田登志 著
A5判・288ページ
定価 3,300円(本体)+税
ISBN 978-4-7741-6709-1

神原健一 著
B5変形判・192ページ
定価 2,580円(本体)+税
ISBN 978-4-7741-7749-6

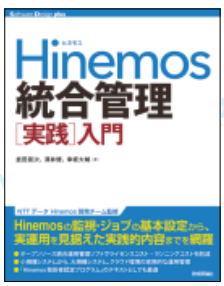


中井悦司 著
B5変形判・200ページ
定価 2,680円(本体)+税
ISBN 978-4-7741-7654-3

上田隆一 著
USP研究所 監修
B5変形判・416ページ
定価 2,980円(本体)+税
ISBN 978-4-7741-7344-3



中島雅弘、富永浩之、
國信真吾、花川直己 著
B5変形判・416ページ
定価 2,980円(本体)+税
ISBN 978-4-7741-7369-6



倉田晃次、澤井健、
幸坂大輔 著
B5変形判・520ページ
定価 3,700円(本体)+税
ISBN 978-4-7741-6984-2

養成読本編集部 編
B5判・176ページ
定価 1,980円(本体)+税
ISBN 978-4-7741-7993-3

養成読本編集部 編
B5判・192ページ
定価2,480円(本体)+税
ISBN 978-4-7741-7858-5

養成読本編集部 編
B5判・128ページ
定価 1,980円(本体)+税
ISBN 978-4-7741-7320-7

養成読本編集部 編
B5判・192ページ
定価 2,280円(本体)+税
ISBN 978-4-7741-7631-4

養成読本編集部 編
B5判・128ページ
定価 1,980円(本体)+税
ISBN 978-4-7741-7607-9

〒162-0846 東京都新宿区市谷左内町21-13
販売促進部 TEL 03-3513-6150 FAX 03-3513-6151

第104回

IT エンジニア必須の 最新用語解説

TEXT：㈲オングス 杉山 貴章 SUGIYAMA Takaaki
takaaki@ongs.co.jp

テーマ募集
本連載では、最近気になる用語など、今後取り上げてほしいテーマを募集しています。sd@gihyo.co.jp 宛にお送りください。

Halium Project

モバイル Linux のためのベースシステムを提供

「Halium Project」は、Android互換のハードウェア上でさまざまなLinuxディストリビューションを実行できるようにする、統一的なベースシステムの開発を目指した新しい取り組みです。

Androidスマートフォンの普及にともない、近年ではさまざまなモバイル端末用のLinuxが登場してきました。おもなものとしては、Ubuntu TouchやSailfish OS、Maemo、Plasma Mobile、Mer、MeeGoなどが挙げられます。しかしいずれのプロジェクトも十分な規模のエコシステムを形成することができておらず、Androidのように大きく普及する気配は見せないまま、開発が中止されたものもあります。

これらのモバイルLinuxは、似たコンセプトを持つものもありながら、基本的には独立したプロジェクトとして開発が進められていました。そのため仕様や実装が共通化されておらず、それが普及の妨げになったという指摘もあります。

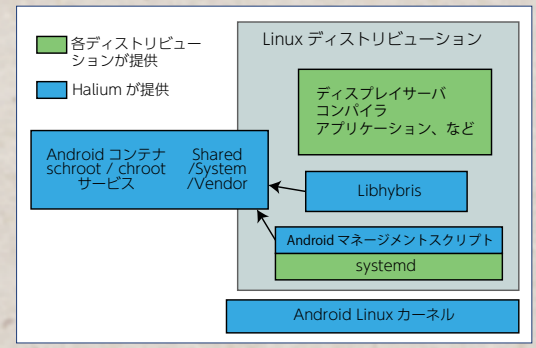
Halium Projectはこの反省を踏まえて立ち上げられたプロジェクトです。Halium Projectでは、Android互換のハードウェア上でLinuxを実行するために共通的に必要となる機能を標準化して提供することを目指します。各ディストリビューションのプロジェクトでは、Halium Projectの成果物を利用することで、OSの基盤となる部分を独自に実装する必要がなくなるため、開発の効率を大幅に上げることができるというわけです。

Halium が提供するスタック

Halium Projectで開発を目指すスタックとしては、次のようなものが挙げられています（図1）。

- Linux カーネル
- Android HAL（ハードウェア抽象化レイヤ）
- Android HAL インターフェース
- Libhybris（Android ライブラリをサポートする互換性レイヤ）
- センサー
- カメラサービス
- RILd（Radio Interface Layer daemon）
- ビルドシステムとスクリプト
- GPS（Mozillaによる AGPS）
- PulseAudio（サウンドサーバ）
- メディアコーデック
- oFono（テレフォニーアプリ開発用プラットフォーム）

▼図1　Halium のスタック概要（公式サイトより引用）



とくに重要なのがLinuxカーネルとAndroid HAL、そしてLibhybrisです。これらはAndroid互換ハードウェアと通信するために必要なスタックであり、従来の主要なモバイルLinuxでも同様のコンセプトが採用されてきました。そこでHaliumではこの部分を標準化し、共通利用できるようにするとのことです。そのほかの部分でも、ハードウェアに近いレイヤで共通化できる部分については、Halium Projectが提供するスコープに含まれています。

その一方で、ディスプレイサーバやツールキット、ユーザインターフェース、アプリケーションなどはHaliumでは提供されない方針です。Haliumの目的はあくまでも共通基盤を提供することであるため、それ以外は各ディストリビューションで独自に実装して提供するべきスタックとされているからです。

Halium Projectはまだ立ち上がったばかりのプロジェクトですが、もし開発に成功すれば、モバイル端末向けの新しいROMの開発が容易になり、モバイルOS市場のさらなる活性化にもつながることが期待できます。Halium Projectでは、UbuntuやSailfishをはじめとするGNU/Linuxベースの OSをサポートすることに加えて、LuneOSやAsteroidOSといった特定機器向けのOSもサポートする予定とのことです。SD

Halium Project
https://halium.org/

DIGITAL GADGET vol.224

安藤 幸央
EXA Corporation
[Twitter] @yukio_andoh
[Web Site] http://www.andoh.org/

音声ガジェット、スマートスピーカーの台頭

3強から スマートスピーカーの登場

スマートスピーカーと呼ばれる、音声入力／出力可能な先進的なスピーカー機器が注目を浴びています。2015年6月にAmazon Echoが米国で179.99ドルで発売されました（日本での発売は未定。Amazon Echo Dotという安価な製品やAmazon Echo Showというタッチスクリーン付きの製品もあり）。2016年11月にはGoogle Homeが米国で109ドルで発売開始となり、日本でも2017年内に発売予定です。そして2017年6月にはAppleがHomePodを349ドルで発表し、年内に英語圏で発売、それ以外の国では2018年になるとのことです。

今年はじめに開催された家電ショーCESでは、Amazon Echoに搭載されている音声システムであるAmazon Alexaが搭載された家電機器が数多く出ていました。こうしてAppleのスマートスピーカーが発表されたことで、ネット系サービスやデジタルサービスの3強からスマートスピーカーが出そろいました。この領域の発展がますます期待されるとともに、競争の場が、パソコンでぽちぽちWebページをサーフィンしていた時代から、スマートフォンアプリに移行し、さらに大きく移り変わっていることが感じられます。

カテゴリとしては「スマート（賢い）」と銘打ち、音声コマンドでさまざまな操作ができ、音や音声、LEDライトなどで状況や様子を知らせたり返答したりする、ネットにつながったマイク兼スピーカー製品です。

各製品の主要機能はクラウドやスマートフォンとの連携が重要になってきますが、各社の思惑はそれぞれ異なります。Appleは音質を重視し、単なるSiri（iPhoneに搭載されている音声アシスタント）デバイスというだけでなく、音楽スピーカーとしても逸品であること。また、Apple MusicやHomeKitによる家電との連動が重視されています。Googleは、Google Play Music、Chromecastほか、グーグルサービスとの連携を進めています。Amazonは、さまざまな機器に組み込めるようにAlexaをAPIとして提供することでサービスの展開を広め、Skillと呼ばれるサードパーティ製の追加機能も拡充されてきています。しかし、Echoは位置情報をもとにサービスが提供されるため、正式に発売されている国以外では不便が多く、国内販売が待たれます。

Amazon Echo

Google Home（提供: Google）

Apple HomePod

※本記事で紹介しているものは
国内未発表・未発売のものを含んでいます。

そのほかにもAndroidの生みの親であるアンディ・ルービンが進めるEssential Home（https://www.essential.com/）、Microsoft Cortana（https://developer.microsoft.com/en-us/cortana）、LINE Clova/WAVE（https://clova.ai/ja）、音響機器メーカーのオンキヨーのAmazon Alexaを搭載した音楽スピーカー、Harman Kordonが発表したCortana搭載のスピーカーなど、スマートスピーカー領域の競争は激しくなってきています。

チャットボットは音声インターフェースの前哨戦

音声認識の精度は上がっており、人間の聞き間違えの割合と同程度の認識率にまでなっています。音声による操作やサービスを考えるための前哨戦として、文字によるチャットボットの設計、デザイン、サービスがたいへん役に立つと言われています。チャットボットでうまくいった対話やサービスのやりとりは、そのまま音声サービスでもうまく機能するからです。

そういった観点で、チャットボットや音声のやり取りのための企画設計と機能デザイン、良い会話のためのテスト観点とは何なのかを列挙してみましょう。

会話編

- 会話ペルソナ（仮想的人物像）を設定しておく。言葉づかいや性格を設定することで会話や個性が表現できる
- 返答までの「間」をうまく設定すること。対応が速すぎても遅すぎても不自然になる
- 100%完璧な認識はないという前提で、話しの流れや、エラーの回避を充分に検討しておく
- そもそも適切な会話や指示をするのは難しい。適切な指示や言葉を提示できると良い
- 延々と会話が続くのではなく、一時停止や、やりとりを終えられるように
- あいまいな表現を避ける。順序だてて会話する。簡単な言い回しで

状態編

- コンテキスト（状態）を引きずり過ぎたり、逆にすぐに忘れてしまったりすると面倒
- 会話の流れによって、同じ回答、同じ言葉が繰り返されないようにすると自然さがでる
- 会話は短く、相手の時間を尊重し、単刀直入に
- 勝手に想像して、口に出していないことを予想して決めつけない
- 重要な事項は、必ず確認を求める

環境編

- 画面がない製品の場合、ON/OFF状態、音声入力を待っている状態を光や音などで示す
- 会話は適切に終了でき、必要のないときはミュート（無音）状態にできること
- 意志のしっかりした人と、優柔不断ですぐに選択ができない人と、さまざまな対話があることを考慮
- 複数の人がしゃべっている状況、雑音の多い環境、必ずしも音響的にベストではないことを考慮

テスト観点

- 特定の言葉を違う言い方で対話してみる
- 言葉や返答が足りなかったとき、どう対応するのかをチェック
- 目的に達するまでの会話の数。会話のやりとりが多すぎても少なすぎても良くない
- 会話の中で、何かを覚えておかなければいけないようであれば、要検討
- 言葉づかい。親しみの度合い。距離感

これからのデバイスとの会話

人間同士の会話は、表情や身振り手振り、その前後のコンテキスト（状況）などをふまえ、さまざまな解釈と伝達が可能です。「このまえのあれ、どうした?」とか「そこのあれ取って!」のような会話が成り立つのは人間同士ならではです。そのうえ、一部でしか通じない業界用語、スラング、方言、省略語、アクセントの間違い、同音異義語、説明不足や早口、不明瞭な発音など、音声操作にはまだまだ対応すべき課

オンキヨー「VC-FLX1」

Harman Kardon「Invoke」

Amazon「Echo Show」

題があります。何も口に出さずとも、先を読んでいろいろなことに対応できれば便利かもしれませんが、意図せずいろんなことが起きてしまうのも困ったものです。

オーストラリア企業のDTの研究所では、音声合成なのか、人による発音なのかを聞き分けるしくみを開発しています。ウェアラブル端末として機能し、人工音声と判断すれば、首の後ろに付けた素子がヒヤッと冷たく感じるのです。たとえば有名人のスピーチの言葉を切り貼りして、全然別な会話を作った場合、声やしゃべり方は本人そのままですが、実際は本人の意図しない会話をさせることも現在の技術であれば可能です。そのために合成音声か本人の発話かを見分ける技術が必要となってくるというわけです。

また、真偽のほどはわかりませんが、Apple Siriの音声合成の品質は、意図的にたどたどしいものとして調整し、Siriの回答に過度に期待をもたせず、少し頼りないものとして、キャラクタ設定をしているとも言われています。音声合成の品質は、技術的には人の会話に近しいところまで来ていますが、声が人間に近すぎると、その対応や会話にも人間と同じ対応が求められてしまうのです。

今後は、コンピュータの認識に適した言語体系、単語、発音の仕方など、言葉そのものが進化してくるかもしれません。短い発音ですべて正しく間違いなく伝達する戦闘用の暗号言語のようなもの、コンピュータ向けのコマンド指示用言語です。たとえば、現状でもAmazon Alexaに認識してもらいやすい、英語の発音講座なども意味のある英語の学び方かもしれません。SF映画「メッセージ(原題:Arrival)」では、思考する言語がその人の考え方や認識を左右するという、サピア＝ウォーフ仮説が紹介されていました。近い将来、人類はスマートスピーカー対応の言語をしゃべりながら、必死で音声操作する夢を見るのかもしれません。SD

Gadget 1

» Lighthouse

https://www.light.house/

スマート監視カメラ

Lighthouseは、一見普通の監視カメラですが、監視カメラで撮影した様子をテキストメッセージに変換して知らせてくれる機器です。ドアが開いたとか、顔認識できない人がいるとかを知らせてくれます。また時系列に従って通知させることもでき、キッチンにペットが入ってきたら通知するとか、誰々が何時までに帰宅しなかったら通知するなどの高度な通知も可能です。299ドルの機器としての販売だけではなく、月額課金のビジネスモデルでサービスを予定しています。

Gadget 2

» Lumigent

https://lumigent.cerevo.com/ja/

音声機能搭載照明

ロボットデスクライトLumigentは、話しかけると変形しつつ点灯／消灯する照明です。単なる照明にとどまらず、カメラを搭載しており、机の上にある書類やメモ、スケッチなどを撮影し記録しておくことができます。撮影も音声コマンドで可能です。照明の位置や明るさなどをいくつか設定しておくことができ、設定した状態は音声コマンドで再現します。また、点灯時の微調整も音声コマンドでできるとのこと。単なる照明というより単機能のロボットとも言える製品です。

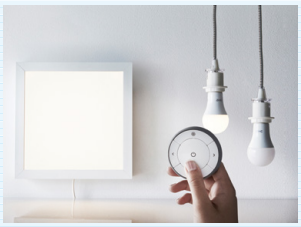
Gadget 3

» TRADFRI

http://www.ikea.com/us/en/catalog/categories/departments/lighting/36812/

音声コントロール照明シリーズ

安価で自分で組み立てる家具を提供するIKEAの照明機器TRADFRIは、AppleのSiri、AmazonのAlexa、Google Assistantに対応したスマート照明機器です。従来はワイヤレスリモコンなどで操作していたTRADFRIシリーズが、音声コントロールできるようになりました。ほかにもスマートフォンでコントロールできる照明機器はいくつか発売されていますが、TRADFRIは製品のバリエーションとともに、そのものが安価であることが大きな特徴です。



Gadget 4

» C by GE Sol

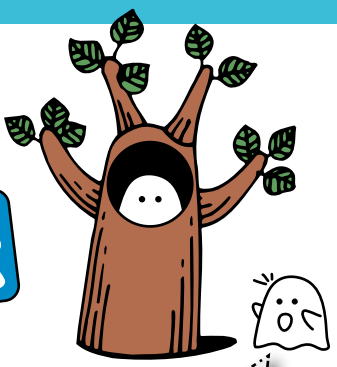
https://www.cbyge.com/pages/sol

スマート照明

C by GE SolはBluetooth対応のスマート照明です。照明機能と、スピーカー、マイクが一体化した製品です。Amazon Alexa機能を搭載し、Amazon Echoと同等の機能を持ったインテリアとしても考えられたスピーカー製品といえます。色温度(照明の色味)や明るさを、音声コマンドか専用のスマホアプリでコントロールできます。照明の光り方で時刻を知らせる機能など、部屋に置いて違和感のない風貌です。

結城 浩 YUKI Hiroshi
http://www.hyuki.com/
Twitter：@hyuki

結城浩の再発見の発想法 51



DRY —— Don't Repeat Yourself

DRYとは

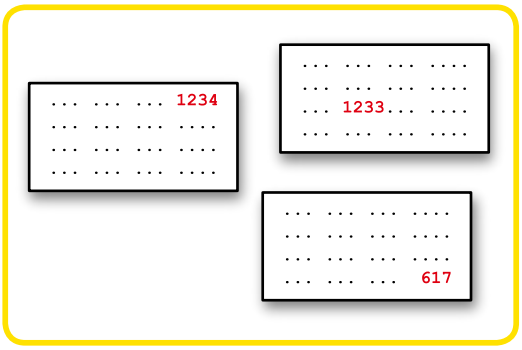
DRYとは、“Don't Repeat Yourself”の略で、**1つのシステム内にある知識は、信頼できてあいまいさがない唯一の表現を持つべきであるという原則**のことです。もっと単純化して「重複を避ける原則」と言うこともあります。DRYは書籍『達人プログラマー』[注1]で詳しく書かれ、Ruby on Railsで採用されて広く認知されている原則ですが、重複を避けるという原則そのものは以前からも存在しました。

知識の重複は、よくトラブルを生みます。たとえば、試験の点数を処理するプログラムを考えましょう。もしも「受験者数」をもとにした数値がソースコードのあちこちに直接埋め込まれていたらたいへんです。受験者数がたとえば1234だとして、1234はもちろんのこと、1を引いた1233や、半数を表す617などが出てくることもあるでしょう（**図1**）。受験者数を変えようと思ったら、ソースコードのあちこちを矛盾なく修正する必要が生じます。

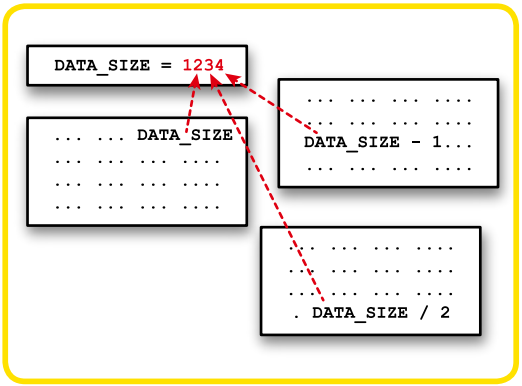
DRYは知識の重複を避けるという原則です。複数個所で知識が必要な場合には、信頼できる唯一の表現を参照します。試験の点数を処理するプログラムの例で言えば、「受験者数」を`DATA_SIZE = 1234`のような定数として、プログラムの1ヵ所で定義します。そして「受験者数」をもとにした数値は必ず`DATA_SIZE`を使って計算します。このようにすれば、プログラム中の矛盾を避け、メンテナンスの手間を軽減できるでしょう（**図2**）。これがDRYから得られるメリットです。

試験の点数がファイルから与えられるなら、DRYをさらに進めることができます。ファイルに含まれるレコード数で「受験者数」を自動的

▼図1　知識が重複している



▼図2　信頼できる唯一の表現を持つ



注1）Andrew Hunt、David Thomas 著、村上雅章 訳、『新装版 達人プログラマー 職人から名匠への道』、オーム社、2016年

に決定できるからです。つまり、プログラマがDATA_SIZEを手で修正するのではなく、与えられたファイルのレコード数をもとにしてDATA_SIZEの値を定めるのです。この場合、受験者数が変わっても、プログラマがDATA_SIZEを修正する必要はなくなります。

別の例として、Ruby on Railsで使われているActiveRecordというクラスがあります。ActiveRecordを使うと、DBスキーマとモデルとの対応関係が自動的に作られるため、プログラマが対応付けの整合性を保つコードを書く必要はなくなります。

プログラムでよく起きるトラブルに、コピー&ペーストによるコードの重複があります。DRYにあてはめるなら、複数個所で同じような処理を実行したい場合には、その**共通処理をまとめる**べきでしょう。似た処理が散らばるのは知識の重複で、似た処理を1つにまとめるのは信頼できる唯一の表現を作っていることになります。

DRYはプログラムだけに適用できる原則ではありません。複雑に絡み合った知識を表現したものならば、どんなものにも当てはまります。たとえば、設計図、技術文書、Webサイト、マニュアル、価格表……複数個所で同じ知識が必要になった場合、それらを個別に書いてはいけません。知識の重複となり、個別にメンテナンスする必要が生まれるからです。

DRYが有効にならない場合

DRYは知識の重複から生まれる矛盾や、メンテナンスの困難を解決します。しかし、DRYは万能ではありません。信頼できる唯一の表現から、必要な知識が自動的に生成されなければ意味がありませんから、そもそも自動化が不可能なシステムでは、DRYを適用することは難しいでしょう。

エラーチェックの場面では、DRYの適用を注意深く行う必要があります。知識の重複という冗長性があるからこそ、エラーチェックが可能になるからです。たとえば、ファイルのレコード数をそのまま受験者数と見なした場合、レコードが欠落するエラーを検出することはできません。冗長なデータであっても「受験者数」を別途用意しておかないと、レコード数と付き合わせたエラーチェックは難しくなります。

DRYを保つためには、システム内にある知識の依存関係を理解する必要があります。知識の依存関係をたどっていき、信頼できる唯一の表現として何を使うべきかを明確にしなくてはなりません。信頼できる唯一の表現を探し出すのが難しいシステムでは、DRYを適用するための労力が異常に高くなる可能性もあります。

日常生活とDRY

日常生活でDRYが必要になる場合はあるでしょうか。

予定表はDRYでないとたいへんですね。たとえば、いつも持ち歩いている手帳に書いてある予定表と、壁に貼ってあるカレンダーに書かれた予定表に矛盾が起きるのはよくあることです。これは2つの予定表という知識の重複が存在するからです。現代ではクラウドを利用した予定表が一般的になったので、PCでも、スマートフォンでも、一元化された予定表を見ることができるようになりました。

組織の指示系統はDRYでないと混乱を生む場合があります。作業者が行う作業について、直接の上司が「作業Aを優先して」と指示を出しているのに、社長が「緊急の作業Bを優先！」と指示を出してはいけません。作業者ごとに、信頼できる唯一の指示者がないと混乱が生じますね。

◆ ◆ ◆

あなたの周りを見回して、複数の知識が存在するためにトラブルを起こしているものはないでしょうか。DRYの原則にしたがって、1つの知識から他方を作り出すことはできないでしょうか。

ぜひ、考えてみてください。SD

及川卓也の プロダクト開発の道しるべ

品質を高めるプロダクトマネージャーの仕事とは？

第10回 バリュープロポジションキャンバス

Author 及川 卓也（おいかわ たくや）
Twitter @takoratta

前回は事業開発手法としてスタートアップのバイブルとなっているリーンスタートアップで用いられるリーンキャンバスを解説しました。今回はその中でもプロダクトマネージャーにとくに重要となるであろう、バリュープロポジション（価値提案）を検討する手法であるバリュープロポジションキャンバスについて解説します。

最初のアイデアの多くは使えない

筆者はハッカソンやその前段階のアイデアソンなどの審査員を務めることがたびたびあります。そのような場で、いろいろとおもしろいアイデアが生まれることもあるのですが、そのまま実用化しても使われないのではないかなと思わせるものも多くあります。

製品やサービスは、もっと言うと事業そのものは、顧客の課題を解決するものか、顧客に新しい価値を提供するものです。アイデアというのは、この課題と課題の解決方法、価値と価値の提供方法を組み合わせたものですが、製品やサービスの企画においてはこれらは分離して考えるべきものです。

「良いアイデアを思いついた！」という人は、そのアイデアにのめり込んでしまっていて、客観的に判断できなくなっていることが多いのですが、本来ならば、まずその課題が本当に顧客が抱えている課題なのか、課題だとしても重要課題なのかを検証する必要があります。考えている課題が確実に重要な顧客課題であることが検証されたならば、次にそれを解決する手段として、考えている手段が最適なのかを検討する必要があります。

本来ならば、このように課題解決や価値提供というのは、2段階のステップを経て行われるべきものなのですが、人はどうしてもそれをセットで考えてしまいがちです。今回、紹介するバリュープロポジションキャンバスはそのような隘路（あいろ）に陥らないようにするためのフレームワークとして有用です。

リーンキャンバスとビジネスモデルキャンバス

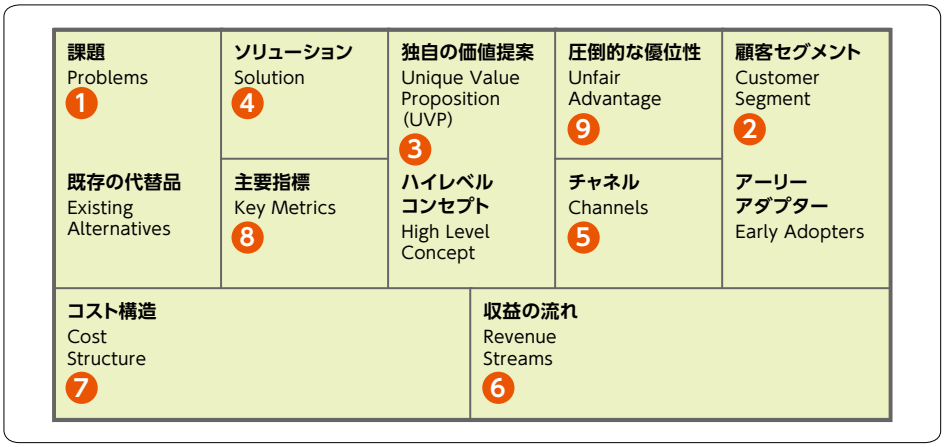
さて、バリュープロポジションキャンバスの話に入る前に、少しリーンキャンバスについて復習しておきましょう。リーンキャンバスはリーンスタートアップの手法を具現化するために、事業モデルを、9つのマスを埋めることで検討するものです（**図1**）。

実はこのリーンキャンバスにはオリジナルがあります。『リーン・スタートアップ』と同じように、こちらもビジネス書としてはベストセラーになった『ビジネスモデル・ジェネレーション ビジネスモデル設計書』[注1]で紹介されているビジネスモデルキャンバスがそれです。ビジネスモデルキャンバスはあらゆるビジネスモデルを俯瞰的（ふかんてき）に表現するフレームワークです。

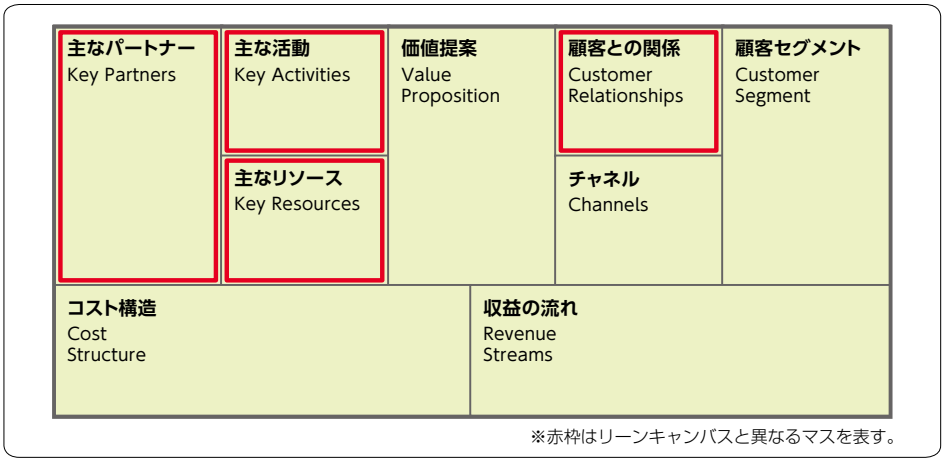
ビジネスモデルキャンバスもリーンキャンバスと同じ9マスによって構成されるキャンバスですが、**図2**に示すように、リーンキャンバス

注1）アレックス・オスターワルダー、イヴ・ピニュール 著、小山 龍介 訳／翔泳社 刊／ISBN978-4798122977

▼図1　9つのマスを埋めることで完成するリーンキャンバス（書籍『Running Lean』より）



▼図2　ビジネスモデルキャンバス（書籍『ビジネスモデル・ジェネレーション』より）



※赤枠はリーンキャンバスと異なるマスを表す。

とはいくつかのマスの内容が異なっています。

この2つのキャンバスは一見すると違いがわからないかもしれませんが、次のように4つのマスが変更されています。

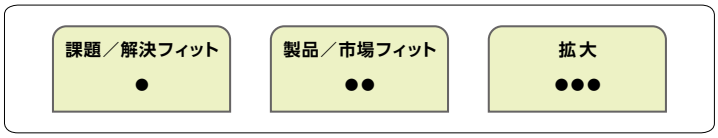
・「主なパートナー」が「課題」に
・「主な活動」が「ソリューション」に
・「主なリソース」が「主要指標」に
・「顧客との関係」が「圧倒的な優位性」に

この変更の理由をリーンキャンバス提唱者のアッシュ・マウリャは「アクションを取りやすい形のキャンバスにするために私がとったのは、もっとも不確かな、より正確には、もっともリスクの高いものを把握するようにすることだった」とブログ[注2]の中で述べています。

確かに、この変更により、リーン開発が一番重要だと定義する、「課題と解決のフィット」と「製品と市場のフィット」（**図3**）がより明確になったと言えるでしょう。

注2）URL https://blog.leanstack.com/why-lean-canvas-vs-business-model-canvas-af62c0f250f0#.2kmwhujo9 より。

▼図3　スタートアップに必要とされる3つのステージ（書籍『Running Lean』より）



バリュープロポジションキャンバス

この課題／解決フィットを検証するために用意されているのが、バリュープロポジションキャンバスです。こちらは『ビジネスモデル・ジェネレーション　ビジネスモデル設計書』と同じ著者たちが書いた『バリュー・プロポジション・デザイン　顧客が欲しがる製品やサービスを創る』[注3]で紹介されています。

このバリュープロポジションキャンバスは**図4**のように、「顧客プロフィール」と「バリューマップ」という2つのパートから構成されています。

顧客プロフィール

顧客プロフィールには、「顧客の仕事」と「ペイン」と「ゲイン」を記入します。

「顧客の仕事」とは、ターゲットとなる顧客が仕事や普段の生活で成し遂げようとしていることです。仕事というといわゆる職業上の職務を想定するかもしれませんが、それに限らず顧客が成し遂げようとしているものがすべて当てはまります。コンシューマがターゲットで、たとえば映画鑑賞のためのサービスを検討している場合には、「気分転換をしたい」とか「好きな俳優を追っかけたい」なども「仕事」となります。

「ペイン」は顧客が「仕事」を達成するに際して障害となるもの、それが起こったならば悪い感情を抱くもの、起こってしまったら不幸になるものなどです。映画鑑賞の例で言うと、「シートが悪く腰が痛くなる」とか「隣の客がうるさい」などがペインとなります。

「ゲイン」は顧客が結果として得られるゲイン（利益）になります。これは予想・期待されていたものもあれば、予想外のものもあります。映画鑑賞の例で言うと、「映画鑑賞後に涙で席を立てないくらいに感動したい」というゲインもあれば、「今まで知らなかった俳優のファンになる」という予想外のゲインも含まれます。

顧客プロフィールへの最後の作業は、「顧客の仕事」と「ペイン」と「ゲイン」に書かれた内容を順位付けすることです。複数の内容がそれぞれ記入されたと思いますので、それらを重要度順にランキングします。これにより、次のバリューマップ作成後のフィットの作業において、本当に重要な課題や価値に対応できているかを確認することが可能となります。

バリューマップ

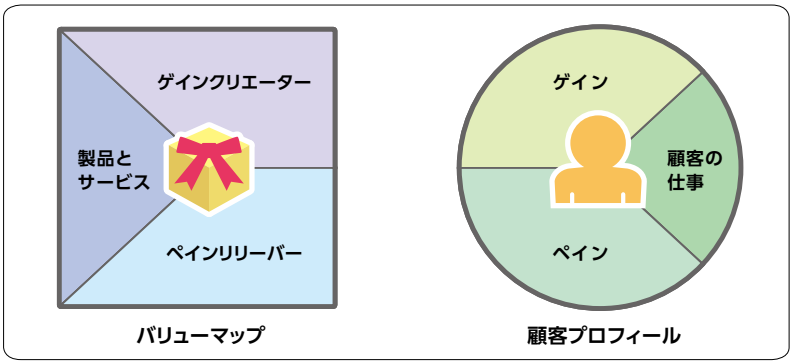
顧客プロフィールの対となるのが、その顧客に対して提供しようと考えている製品・サービス側を可視化したバリューマップです。

バリューマップには、「製品とサービス」と「ペインリリーバー」、「ゲインクリエーター」を記入します。

「製品とサービス」は提供する製品やサービスです。製品やサービスは複数により構成されることがほとんどです。たとえば、映画鑑賞のためのプレミアムシアターを考えていたとすると、「会員限定の上質の映画館」

注3）アレックス・オスターワルダー、イヴ・ピニュール、グレッグ・バーナーダ、アラン・スミス 著、関 美和 訳／翔泳社刊／ISBN978-4798140568

▼図4　顧客プロフィールとバリューマップ（書籍『バリュー・プロポジション・デザイン』より）



というものに加えて、「オンラインサロン」や「会員特典としての特別上映」などが考えられます。

「ペインリリーバー」は製品やサービスで課題(顧客プロフィールで言う「ペイン」)を解決する方法を書きます。たとえば、「映画鑑賞後のフィードバックとして周りの客のレーティングを行い、レーティングがあるレベルより低くなった会員は会員権を剥奪(はくだつ)する」ことや「人間工学に基づくシートの提供」などが考えられます。

「ゲインクリエーター」には価値の提供方法を書きます。たとえば、「映画の雰囲気に合わせた装飾を施すことで映画鑑賞の前から気分が盛り上がり、また鑑賞後もその余韻に浸れるようにする」ことや「ミニシアター系の映画も取り上げ、セレンディピティ(偶然に思わぬ幸運をつかむこと)を起こしやすくする」などが考えられます。

冒頭で、製品やサービスは顧客の課題を解決するか、顧客に価値を提供するものであると書きましたが、ここではその解決方法や価値提供手段を書きます。

このバリューマップもこの3つの項目内で順位付けを行います。

顧客プロフィールとバリューマップによるフィットの検証

顧客プロフィールの説明をした後ですので、もしかしたら顧客プロフィールに書いた「ペイン」や「ゲイン」の解決方法や提供方法が書かれるだけかと思われるかもしれませんが、バリューマップは顧客プロフィールとは独立して書く必要があります。

独立して書かれたこの2つの、とくに「ペイン」と「ペインリリーバー」、「ゲイン」と「ゲインクリエーター」を比較することで、自分たちの製品やサービスが顧客の課題を本当にどれだけ解決できているか、顧客の期待する価値をどれだけ提供できているかがわかるのです(**図5**)。

リーンキャンバスの中でのバリュープロポジションキャンバスの利用

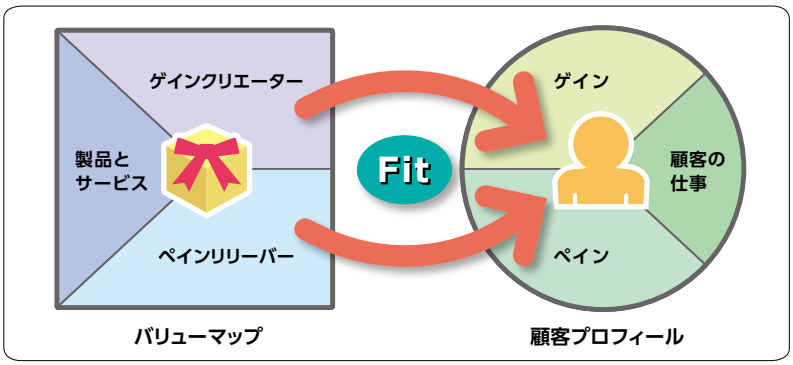
以上がバリュープロポジションキャンバスの説明となります。説明の中では「記入する」とか「記述する」と書いていますが、実際には出てきたアイデアをグルーピングしたり、順位付けしたりするときに並べ替えたりするので、付箋紙(ふせんし)で作業を進めることを推奨します。

このバリュープロポジションキャンバスでの作業は、リーンキャンバスの作成の中では、前半の❶から❹までのマスでおもに使うことになります(**図1参照**)。このバリュープロポジションで課題／解決フィットを検証し、次にMVP(Minimum Viable Product)などで製品／市場フィットを確認していくことになります。

大事なのはどの段階においても、仮説と検証の反復を行い、必要に応じて方針変更(ピボット)をしていくことです。

今回はバリュープロポジションキャンバスを用いた課題／解決フィットについて説明しました。このキャンバスは仮説の段階で一度作成し、そのあと、その仮説の検証後に再度作成します。さて、次回は少し話題を変えて、米国で行われるプロダクトマネージャーのカンファレンスであるMind The Productについて紹介したいと思います。SD

▼図5　顧客プロフィールとバリューマップで検証する課題／ソリューションフィット



Profile
ソフトウェアエンジニアとして社会人キャリアをスタートした後、MicrosoftやGoogleでプロダクトマネージャーやエンジニアリングマネージャーを経験。現在はいくつかのスタートアップのプロダクトマネージメントをサポートしている。

宮原徹の

オープンソース放浪記

第18回 OSC名古屋とご当地カントリーマアム

宮原 徹（みやはら とおる） Twitter @tmiyahar 株式会社びぎねっと

尾張名古屋は手羽先でもつ

今年度のOSCシーズンが始まりました。今回は5月27日（土）に開催されたOSC名古屋、そして9月2日（土）に開催予定のOSC千葉の会場下見の様子などをお送りします。

学生スタッフが支えてくれたOSC名古屋

OSC名古屋は1日で500名が参加する、全国の中では比較的大規模な開催となる地域です。そのため、これまで会場が名古屋市立大学、名古屋国際センター、そして昨年からは吹上ホールと、より大きな会場へと変更になっています。ちなみに吹上ホールは展示即売会が行われているような施設で、今回もお隣ではピアノの展示販売が行われていました。

会場が大きくなってゆったりしていいのですが、その分準備に手間がかかります。毎回、スタッフ確保に苦労するのですが、今回は「東海道らぐ」「SWEST & LED-Camp」「中部大学」「名古屋工業大学」「名古屋市立大学」など多数のご協力で、前日準備や当日運営を行えました。アルバイトなどで人手をまかなうのは簡単なのですが、こうやっていろいろな方、とくに学生さんたちに運営を支えてもらうことで、この経験が今後何かの役に立てばいいなと思います。前日準備終了後は、みなさんを労うために会場近くの「元祖手羽先 風来坊」で手羽先を堪能しました（**写真1**）。

▼写真1 風来坊の手羽先。美味しかったので2回も3回もおかわりすることに

懇親会の料理が瞬殺に

OSC名古屋（**写真2**）終了後の懇親会での出来事です。私は荷物がすべて搬出されて、会場を返却してから懇親会会場に向かったのですが、途中で「料理が食べ尽くされました」とのメッセージが届きました。予約時より多少人数は増えましたが、たった15分で料理が終わってしまうのは尋常ではありません。大急ぎで現場に急行しました。原因は、事前にやりとりをしていたお店側の担当者が不在で、事前打ち合わせの内容が当日の担当者に伝わっていなかったこと。現地で幹事を務めてくれた方が、お店側と交渉してアレこれと追加してもらえましたが、参加していただいたみなさんには少し不満が残ってしまったかもしれませんね。

OSCのようなイベントだと、なぜか立食形式の懇親会では料理がすぐになくなってしまうことが多いです。

▼写真2 OSC名古屋の受付スタッフは、名古屋市立大学の学生で、博物館の企画をサポートするボランティアサークルMAROのみなさん

▼写真3 飲み足りない面々は2次会で多いに盛り上がりました。名古屋の夜はまだこれからです

▼写真4
小倉トースト風味。実はOSC名古屋の翌日に開催されていたAndroidイベントへの差し入れだったりします

▼写真5　千葉工業大学の会場下見。机の並べ方などを頭の中でシミュレーションしていきます

これがビジネス系の懇親会ですと料理は余り気味になるのはなぜでしょう。対策として、おやつなどを出して空腹感を紛らわす、とにかく炭水化物を多めにするなどがありますが、根本的な解決は難しいものがあります。このあたりがイベント運営のおもしろさでもあるのですけどね。

結局、物足りない連中は、会場からのシャトルバスで最寄り駅まで送ってもらったあと、2次会(**写真3**)で食べ、さらに3次会で飲み、名古屋の夜は更けていくのでした。

ご当地カントリーマアム

いつからか、「OSCといえばカントリーマアム」というツイートをするようになりました。もともとは運営スタッフ用に、美味しく、個包装なので余っても困らないので用意をしていたのです。最近では、「ご当地カントリーマアム」をお土産や差し入れとしていただくようになりました。そういえば前回のOSC名古屋のときには、大阪から参加してくれた高校生が「神戸プリン風味」のカントリーマアムを差し入れてくれました。

今回のご当地カントリーマアムは、名古屋といえば「小倉トースト風味(**写真4**)」。「お味は？」というと、普通に小倉味で美味しかったです。

OSC千葉の会場は千葉工業大学

昨年はアンカンファレンス形式で開催した千葉ですが、今年は若干ステップアップして150名の大教室とブース展示での開催となります。開催は千葉工業大学で9月2日(土)です。「OSC東京はちょっと遠くて」という千葉県民のみなさん、ぜひ当日はご参加ください。

どのような形で開催できるか検討するため、また実行委員会で意見交換を行うため、会場を下見してきました(**写真5**)。

下見には何名か在学中の学生さんに参加してもらえました。当日、単なるお客さんとして参加するのではなく、企画段階から入ってもらえることで何かを学び取ってもらえればと思っています。

会場は教室も広いですし、展示スペース用の廊下部分もゆったりしているので、当日はじっくりとセミナー、ブース展示見学をしてもらえそうです。開催が楽しみですね。SD

Report

大阪、京都と関西方面も盛り上がってきました

千葉会場下見の翌日は大阪へ出張です。大阪での開催は今年も来年も1月ですが、「この間も実行委員会中心に少しずつでも集まろう」ということで、LT大会を企画してもらいました。当日は前回の実行委員だけでなく、新しい人も参加して、ビール&ピザでお互いのLT発表を楽しみました。来年1月の開催までさらに何回か開催して結束を強めていきたいところです。

その後は京都に移動して、8月開催に向けての決起集会です。実行委員だけでなく、ITとはまったく関係ない人たちにも声をかけて参加してもらい、いろいろと話を聞きました。「WordPressでWebサイト作りを始めてみたい」という意見が出たので、さっそくOSC京都ではWordPress入門のセミナーを開催してみようということになりました。このような形で、どんどんニーズを満たすセミナーを企画していきたいですね。

さて、この文章は沖縄行きの飛行機の中で書いています。次回はOSC沖縄の様子をレポートしたいと思います。もちろん、オリオンビールと泡盛も。

◀さくらインターネットの林 雅也氏の紙LT。少人数ならこれもアリですね

▶京都での決起集会の様子。すでにグッスリと寝てしまっている人も？

ツボイのなんでもネットにつなげちまえ道場

第26回

Groveを使ってみる

Author 坪井 義浩（つぼい よしひろ） Mail ytsuboi@gmail.com @ytsuboi
協力：スイッチサイエンス

はじめに

そういえば、Grove（グローブ）のことを本連載でまだ扱ったことがなかったので、今回は手軽にマイコンと電子部品を接続することのできるGroveを紹介したいと思います。電子工作やIoT機器のプロトタイプをするとき、電子回路や基板の設計、あるいははんだづけをすることがハードルになっている方も多いと思います。

Groveは、中国の深圳（シンセン）にあるSeeed社が提供しているシステムです。さまざまなGroveのモジュールが提供されており、Groveのケーブルをコネクタに差し込むだけで、簡単にこれらモジュールとマイコンを接続できます。マイコン基板にGroveのコネクタが付いていない場合、「Base Shield（**写真1**）」を使ったり、「Grove to 4Pin Male」というケーブル（**写真2**）を使ったりして接続します。

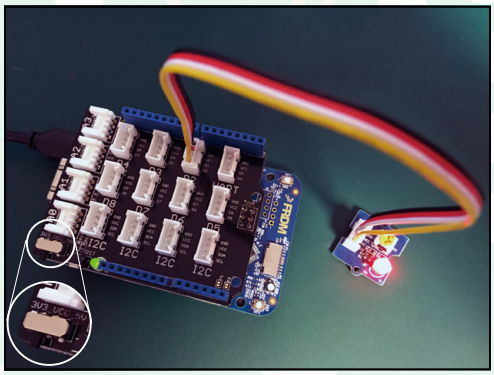
Groveは、GPIO（汎用入出力）のほかに、UARTやI²C、アナログといった信号の接続にも対応しています。ただ、GPIOもほかの信号もすべて同一のコネクタですので、接続するときには信号の種類に注意しなければなりません。

もうひとつ、Groveはもともと Arduinoのために作られた規格です。そのため5Vが前提になっていましたが、今ではmbedやほかのマイコンボードでも使うために3.3Vの電源や入出力も混在しています。この点にも注意しなければいけないのが、Groveの少々面倒なところです。

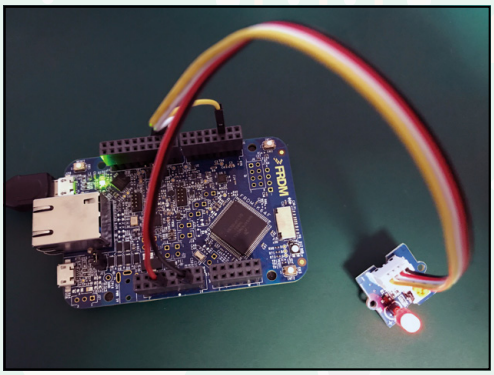
はじめのLED点滅

さて、まずFRDM-K64FにGroveのLEDを接続して点滅させてみましょう。今回は「Base Shield V2」を使って「Grove - LED」をFRDM-K64FのD2に接続します。これらは筆者の手元にあったGrove Starter Kitから取り出して使いました。Starter Kitに入っているGrove LEDは、LED Socket Kitという LEDを差し込むソケットのついたGroveモジュールと、青・緑・赤の3つの砲弾型LEDが入っていました（**写真3**）。今回は、赤い砲弾型LEDのリード線を短く切りそろえ、LEDの樹脂部分の切り欠きと基

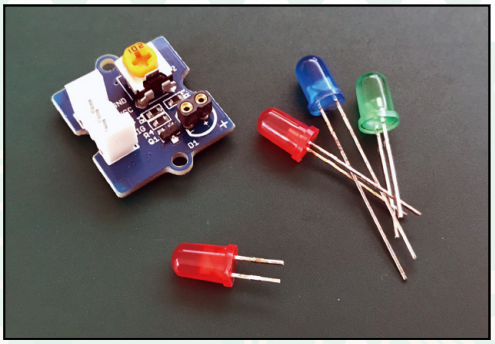
▼写真1　Base Shieldの使用例



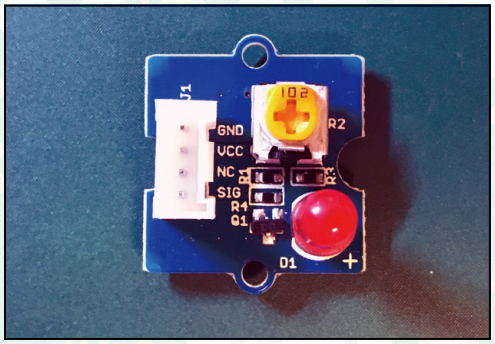
▼写真2　Grove to 4Pin Maleケーブルの使用例



▼写真3　組み立て前のモジュール



▼写真4　モジュールを組み立ててみた



板の印刷の切り欠きの方向を合わせて挿し込みます(**写真4**)。

LEDモジュールを組み立てたら、FRDM-K64FにBase Shieldを取り付け、GroveのケーブルでBase ShieldのD2と表示されたコネクタとLEDモジュールを接続します。Base Shieldの端についているスライドスイッチを、基板上の3V3(3.3V。「.」は読みづらいのでわかりやすいように「3V3」と表記(**写真1左下**)することがあります)側に設定します。このスイッチは、Base Shieldについているグローブコネクタから供給する電源の電圧を選択するものです。FRDM-K64Fの入出力は3.3Vですので、Groveのモジュールも3.3Vで動かします。これでハードウェアの準備は完了です。

ソフトウェアは、mbedのオンラインコンパイラを開き、「新規」ボタンをクリック、プラットフォームにFRDM-K64Fを選択したうえで「mbed OS Blinky LED HelloWorld」を選択して「OK」ボタンをクリックしてください。LED

▼リスト1　LED点滅プログラム(main.cpp)

```
 1 #include "mbed.h"
 2
 3 DigitalOut led1(D2);  ←ここを変更
 4
 5 // main() runs in its own thread in the OS
 6 int main() {
 7     while (true) {
 8         led1 = !led1;
 9         wait(0.5);
10     }
11 }
```

の点滅ですので、「プログラムとライブラリを最新のバージョンにアップデートする」にチェックを入れておいても大丈夫でしょう。今回は、D2にLEDを接続したので、**リスト1**の3行目のLEDを接続しているピンをD2に変更しました。

コードをコンパイルしてmbedに書き込み、リセットボタンを押すと**写真1**のようにLEDが点滅します。

超音波距離センサ

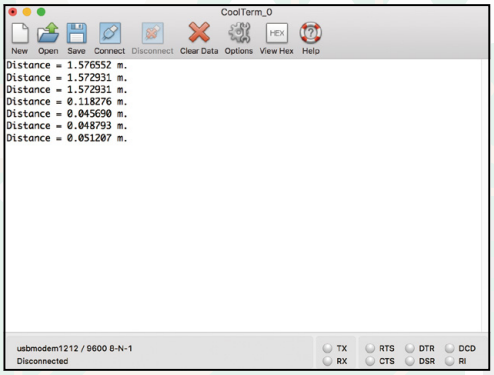
今度はセンサを使ってみましょう。手元に「Grove - Ultrasonic Ranger(**写真5**)」(超音波距離センサ)[注1]がありましたので、今回はこれを使ってみます。冒頭に記したように、Groveを使うときには対応している電圧を確認しなければなりません。先ほど記したようにFRDM-K64Fの入出力は3.3Vですので、3.3Vに対応したGroveモジュールでなければ正常に動作しな

注1)　http://wiki.seeed.cc/Grove-Ultrasonic_Ranger/

▼写真5　Grove - Ultrasonic Ranger

▼写真6　Grove - Ultrasonic Rangerの説明書き

▼図1　シリアルターミナルへの出力例



いでしょう。たとえば今回の超音波距離センサの場合、パッケージに入っていた紙(**写真6**)にも対応する電圧が3.3V/5Vと記されていますので使用できそうです。

超音波距離センサのmbedのソフトウェアも、mbed.org[注2]で公開されています。超音波距離センサは、超音波を送信してから戻ってくるまでの時間を利用して距離を測ります。この超音波距離センサは、デジタル信号1つで接続できるようです。公開されているサンプルコード(**リスト2**)の27行目では、D0を使うように指定されていましたが、D0とD1はArduinoフォームファクタ(Arduino Unoと同じピン配置のソケットをこう呼びます)ではUARTに使われていることもあるので、今回はD4に接続してみました。

ハードウェアの用意は簡単です。Base Shield

注2)　https://developer.mbed.org/components/Grove-Ultrasonic-Ranger/

▼リスト2　超音波距離センサのmbedのソフトウェア(main.cppの22行目から抜粋)

```
22 #include "mbed.h"
23
24 #include "RangeFinder.h"
25
26 // Seeed ultrasound range finder
27 RangeFinder rf(D0, 10, 5800.0, 100000);  ←この行のD0をD4に変更
28 DigitalOut led(LED1);
29
30 int main()  {
31     led = 1;
32     float d;
33     while (1)   {
34         d = rf.read_m();
35         if (d == -1.0)  {
36             printf("Timeout Error.?n");
37         } else if (d > 5.0) {
38             // Seeed's sensor has a maximum range of 4m, it returns
39             // something like 7m if the ultrasound pulse isn't reflected.
40             printf("No object within detection range.?n");
41         } else  {
42             printf("Distance = %f m.?n", d);
43         }
44         wait(0.5);
45         led = !led;
46     }
47 }
```

のD4と書かれたGroveコネクタと超音波距離センサをGroveのケーブルでつなぐだけです。Base Shieldの端についているスライドスイッチが3V3側にセットしてあるのも確認しておきましょう。

ソフトウェアも簡単でした。サンプルコードをインポートし、main.cppの27行目のD0をD4に変更し、ターゲットがFRDM-K64Fであることを確認してコンパイルします。ダウンロードされたバイナリファイルをmbedのドライブにドラッグ＆ドロップして書き込みましょう。

書き込んだら、適当なシリアルターミナルを立ち上げ、mbedのシリアルポートを開きます。今回使っているサンプルコードはmbed 2で作られていますので、「9,600bps、8bitパリティなし」で開いてください。そこでFRDM-K64Fのリセットボタンを押すと、シリアルターミナルに**図1**のように超音波距離センサで測定した距離が表示されます。**写真7**のように超音波センサを上向きに置いて動かしてみたので、最初の3行くらいは天井までの距離が表示されていた模様です。4行目から急に短い距離が表示されているのは、手をかざしたからです。

Grove

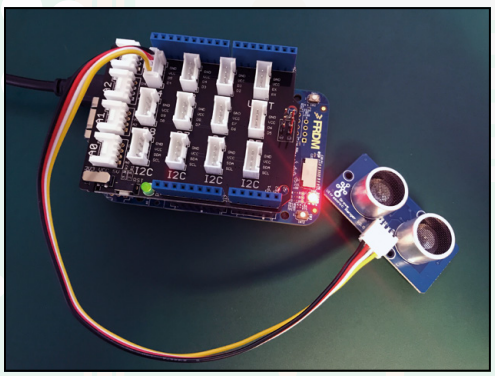
このように、Groveを使うと簡単にセンサなどのデバイスを接続して、手軽に試作を行うことができます。Groveのモジュールは100種類以上あります。**写真8**はSeeed社によるGroveを紹介する展示です。今回使ってみた超音波距離センサのほか、さまざまなモジュールがあることが見て取れます。モジュールのみならず、Groveを接続するためのコネクタを搭載したマイコンは、mbed以外にもArduino互換機など、さまざまな製品があります。

Groveコネクタが搭載されていないマイコンボードには、Arduinoフォームファクタであれば、今回使ったBase Shieldを使用すれば簡単にGroveコネクタを追加できます。この連載でも紹介してきたmbed LPC1768にGroveコネクタを追加するには、mbed Shield(**写真9**)[注3]を使用するのが手軽な方法です。この基板を使えば、Groveコネクタ、mbed LPC1768のUSBやEthernetコネクタだけでなく、Arduinoフォームファクタのソケットも追加できます。SD

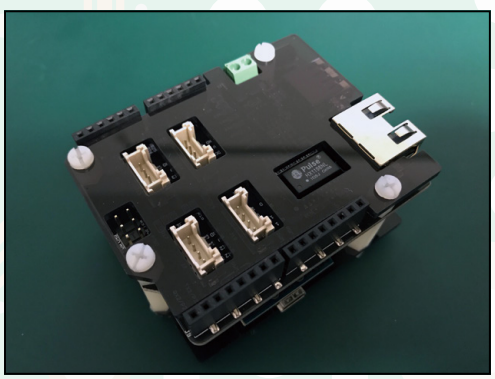
注3) https://www.seeedstudio.com/mbed-Shield-p-1390.html

▼写真8　Seeed社の展示

▼写真7　超音波距離センサを動かしてみた



▼写真9　mbed Shield



読者プレゼントのお知らせ

『Software Design』をご愛読いただきありがとうございます。本誌サイト**http://sd.gihyo.jp/**の「読者アンケートと資料請求」にアクセスし、アンケートにご協力ください（アンケートに回答するにはgihyo.jpへのお名前と住所のアカウント登録が必要となります）。ご希望のプレゼント番号を記入いただいた方の中から抽選でプレゼントを差し上げます。締め切りは**2017年8月17日**です。プレゼントの発送まで日数がかかる場合がありますが、ご容赦ください。

ご記入いただいた個人情報は、プレゼントの抽選および発送以外の目的で使用することはありません。アンケートの回答は誌面作りのために集計いたしますが、それ以外の目的ではいっさい使用いたしません。記入いただいた個人情報は、作業終了後に責任を持って破棄いたします。

01 SSD「S300TL240K」 1名

厚さ7mm・サイズ2.5インチ・容量240GB・SATA 6Gbps対応のSSDです。SiliconMotion社（SMI）製コントローラとTLC NANDフラッシュメモリを採用し、NCQ/TRIMにも対応した、最大読込540MB/s、最大書込490MB/sのハイパフォーマンスモデルとなっています。9.5mm厚への変換スペーサーも付属します。

提供元 マスタードシード http://www.mustardseed.co.jp

02 SIMPLE SMART BOX 1名

2.5インチSATA HDD/SATA SSD用のケースです。7mm/9.5mm厚に対応しています（7mm厚用に固定クッションが付属）。ホストマシンとは付属のケーブルを使ってUSB3.0で接続します。新色フローズンブルーをプレゼント。

提供元 センチュリー http://www.century.co.jp

03 Octocatぬいぐるみ 1名

6月6日に開催された、GitHubの公式イベント「GitHub Constellation Tokyo」で販売された、GitHubのマスコットキャラクタ「Octocat」のぬいぐるみです。ノベルティのハンドバッグやステッカーとセットでプレゼントします。

提供元 ギットハブ・ジャパン https://github.co.jp

04 人狼知能で学ぶAIプログラミング 2名

狩野 芳伸 ほか 著

心理的な駆け引きが楽しめるテーブルトークRPG「人狼」。その人狼ゲームをAI同士で行う「人狼知能エージェント」を作りながら、AIプログラミングを学べる1冊です。使用プログラミング言語はJavaです。

提供元 マイナビ出版 http://pub.mynavi.jp

05 動かして学ぶセキュリティ入門講座 2名

岩井 博樹 著

マルウェア、スパイウェア、ランサムウェアといったセキュリティ攻撃のしくみをひも解きながら、プロのセキュリティ技術者が現場で使用するツール類で攻撃の検知やシステムの防御を行う術を解説します。

提供元 SBクリエイティブ http://www.sbcr.jp

06 プログラマのためのGoogle Cloud Platform入門 2名

阿佐 志保、中井 悦司 著

IaaSサービス「Google Cloud Platform」の入門書です。サービスの全体像の紹介から、Webアプリの実行基盤とコンテナ実行環境の作り方、PaaSサービス「Google App Engine」の使い方まで解説します。

提供元 翔泳社 http://www.shoeisha.co.jp

07 Webフロントエンドハイパフォーマンスチューニング 2名

久保田 光則 著

Webサイト、Webアプリを高速にチューニングするための解説書。ブラウザのレンダリングから、個別の問題に対する対応例、今後を見据えた設計の基礎など、その場しのぎではない高速化方法を学べます。

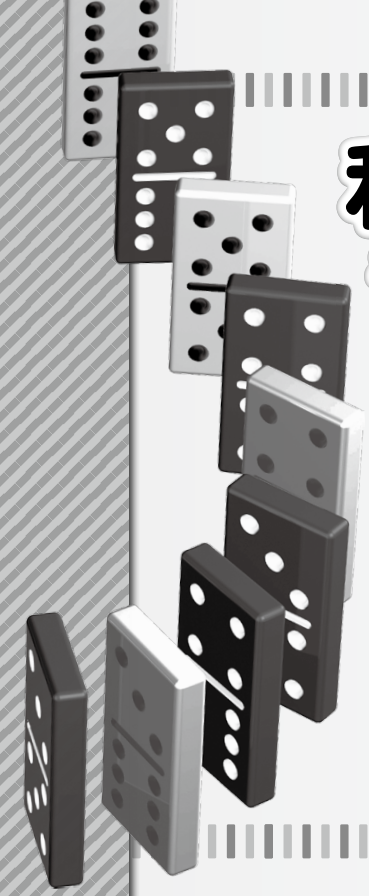
提供元 技術評論社 http://gihyo.jp

START!

第1特集

私も機械学習エンジニアになりたい!

先端Web企業の取り組み方は?



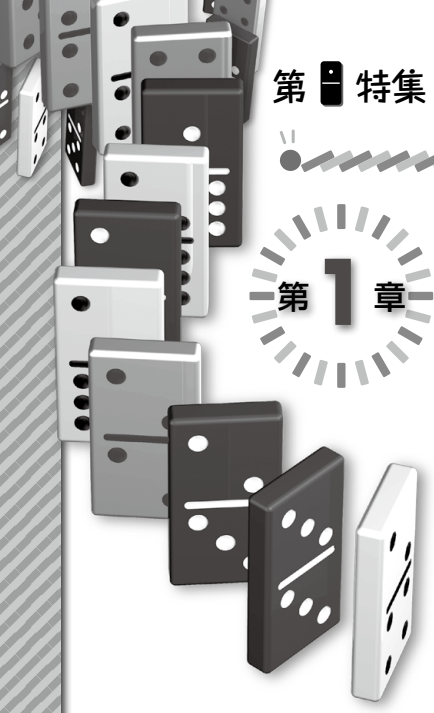
機械学習は幅広く、奥深い技術です。

いざ機械学習の勉強を始めるとなると、どこから手を付けてよいのかわからないかもしれません。

そこで本特集では、機械学習の全体像をつかむために第1章を、ディープラーニングの基本的な考え方を理解するために第2章を、GPU搭載の機械学習用PCが欲しくなったときのために第3章を、現場で求められている人材や知識の参考に第4章をご用意しました。

ここが機械学習エンジニアへとつながるスタートラインです。

GOAL!



第1章

ビジネスの変革をもたらす機械学習

深層学習、人工知能との違いとは

本章では、最近ビジネスに活用されている機械学習、深層学習、人工知能について概観します。それらがビジネスにどのように活用されているか、またその際に誤解されやすい事柄を整理します。まずは本章を、機械学習をビジネスに応用するうえでの足がかりにしてもらいたいと思います。

Author 黒柳 敬一（くろやなぎ けいいち）Sansan㈱Data Strategy & Operation Center
Twitter @Keiku Blog http://keiku.hatenablog.jp/
Web https://www.kaggle.com/keiku322

機械学習とは？

機械学習は、もともと人間の持つ学習能力を機械（計算機）に持たせることを目指す人工知能の一研究分野として発展してきました。機械学習とは一言で言うと、「機械（計算機）によってデータから何らかの法則（ルール）を学習（ルールを構築）し、その法則によって決められたパターンを得ること」です。

機械学習でできること

機械学習でできることは大きく分けて2つあります。それは、「予測」と「発見」です。冒頭で触れた得られるパターンというのは、これらを指します。この予測と発見は、それぞれ用いるデータの期間や目的などが異なります。

- 予測：過去から現在までのデータをもとに未来／未知のデータに対して予測する
- 発見：過去から現在までのデータから未知のパターンを発見する

予測は、機械学習の枠組みとしては「教師あり学習」と呼ばれます。学習において予測のお手本となる過去の正解データを用いて学習することから、このように呼ばれます。教師あり学習によっておもに、「分類」と「回帰」というタスクを実行できます。

分類というのは、過去のデータの分類傾向に基づいて未来のデータではどのように分類されるかを予測する方法です。一方で回帰というのは、過去のデータの連続値の推移から、未来のデータがどのような値になるかを予測する方法です。とくに時系列的に未来のデータに限らず、未知のデータに対して予測できます。

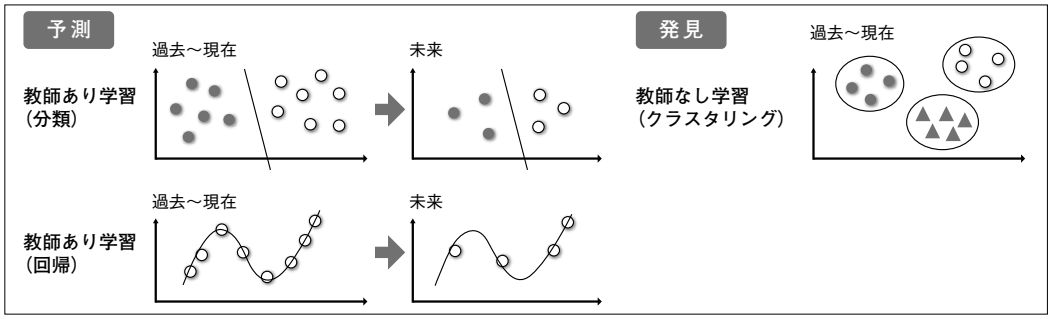
また発見は、機械学習の枠組みとしては「教師なし学習」と呼ばれます。教師あり学習とは異なり、学習において過去の正解データを用いることなく学習することから、このように呼ばれます。教師なし学習では、おもに「クラスタリング」というタスクを実行できます。クラスタリングというのは、似ているものをまとめることで新たな知見を得る方法です。

これらを概念図として表すと**図1**のとおりです。

今ビジネスで活用されている機械学習

機械学習の分類や回帰、またクラスタリングの手法は、さまざまな領域における課題に対して適用できます。そのためには、まずコンピュータが学習するための「データ」が必要です。データさえあればさまざまな課題に対応できるようになります。機械学習の適用事例を、領域ごとに**表1**に示します。これらの事例からもわかるように、金融、マーケティング、Webなどさ

▼図1　機械学習でできること



まざまな領域で機械学習が活用されていることがわかります。

さまざまな領域で活用されている背景としては、「ビッグデータ」の重要性が叫ばれる中、その蓄積と活用のためのインフラが整えられてきたことが大きな要因と言えます。また、「オープンソースソフトウェア（OSS）」の発展の寄与も大きく、機械学習を容易に使うことができるようになったことも理由の1つです。

機械学習による学習と予測

ここで機械学習における学習と予測についてイメージを持ってもらうために、やや正確性を犠牲にして直観的に解説します。

機械学習による学習は、「直線を引くイメージ」です。何か2つのクラスで分けられていることが知られているデータがあったとします。このデータは、たとえば、腫瘍（しゅよう）の悪性／良性、ECサイトでの購入あり／なし、迷惑メール／正常メールなどの区分です。このデータに含まれる特徴を軸にとり、**図2**のようにプロットしてみます。たとえば、腫瘍の例であれば、腫瘍のサイズ、人の性別、年齢などが特徴にあたります。このデータの2つのクラスをうまく分けられるような直線を引くことを考えます。

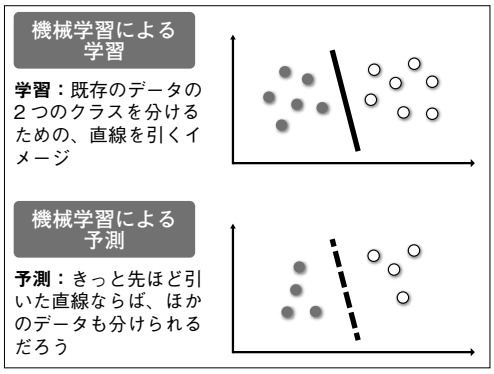
大雑把に言えば、この直線を自動で引くしくみが機械学習の「学習」です。具体的には、直線に一番近い黒丸／白丸双方への距離（マージンと言います）がなるべく大きくなる直線を引けるように学習します。この引かれた直線が別のデータでもきっと同じように分類できるだろうと思って予想して、別のデータにも直線を引くことが機械学習の「予測」です。これが機械学習における学習と予測のイメージです。

この学習と予測に利用できるアルゴリズムはさまざまあり、決定木、ロジスティック回帰、ランダムフォレスト、ニューラルネットワーク

▼表1　機械学習の事例

領域	機械学習の事例
金融	市場予測、与信審査における信用リスク評価、不正検知
マーケティング	需要予測、顧客セグメンテーション、ダイレクトメール（DM）のターゲティング、評判分析
Web	情報検索、スパムフィルタリング、レコメンデーション、機械翻訳、ソーシャルネットワーク分析
広告	広告のクリック率予測
ヘルスケア	MRI画像による医療診断
マルチメディア	音声認識、画像認識
機械・製造業など	故障・異常検知

▼図2　機械学習による学習と予測



など多数存在します。これらのアルゴリズムについてはどこかで耳にしたことがあるかもしれません。それぞれのアルゴリズムについての解説は、誌面の都合上、ほかの書籍に任せたいと思います。参考文献[1][2]を参照してください。

機械学習の性質を知る

表1で取り上げた機械学習の事例は、実はどれも人が判断できる法則ばかりです。ここで、人と比較することで機械学習にはどのような特徴があるのか説明します。

人と比較して機械学習の得意なところ

人と比較して、機械学習には次のような有利な点があります。

- 人手で処理できる量以上に「大量に処理」できる
- 人の処理速度以上に「高速に処理」できる
- 人が判断するよりも「高精度に判断」できることがある

しかも、コンピュータは疲れることを知りません。これらの利点が大きな要因となり、機械学習は実世界においても利用されています。

日常の生活でも、機械学習による大量・高速な処理の恩恵を実感できます。たとえば、迷惑メールを除外する機能であるスパムフィルタリングがあります。もしこの機能がなければ、途方もない数の迷惑メールを人手でひとつひとつ閲覧してゴミ箱に捨てなければなりません。しかし機械学習を用いれば、容易に大量・高速に迷惑メールを除外できます。

また、機械学習が高精度に判断できる最近の事例として、Facebook社によるDeepFaceという人間の顔認証技術が、人と同等以上の顔認証の精度を実現しています。特定の分野については、人の判断や処理に要する時間やその精度を上回っていることは明らかです。とくに「深層学習」と呼ばれる機械学習の技術が、高精度に判断できる要因となっています。このような背景もあり現在、深層学習の研究開発が盛んに行われています。

人と比較して機械学習の苦手なところ

逆に、人と比較して機械学習を用いるには難しい場面もあります。

- 「少ない情報から推論する」ことが得意ではない
- 状況の変化に対して柔軟に対応する

データが少ないと機械学習を利用することは難しいでしょう。人は少ない情報から推論することが得意ですが、これは機械学習では苦手な処理です。冒頭でも触れましたが、やはりデータが大量にあることで機械学習は力を発揮します。

また機械学習のロジックは、アルゴリズムの組み合わせでしかないので、ルールに当てはまらないような処理に対応することが苦手です。たとえば、再びスパムフィルタリングの例を考えてみましょう。スパムフィルタリングも万能ではなく、度々、人が正常であると判断されるメールについても迷惑メールフォルダに振り分けられていることがあると思います。この問題が生じる理由としては、過去のデータから学習したルールに当てはまらない事象が生じてしまったことが原因となっています。

深層学習とは？

以下、やや正確性に欠けるかもしれませんが、深層学習を簡単に理解してもらうために、機械学習における深層学習の位置づけと、簡単な歴史を解説します。より詳しい解説は参考文献[3][4]を参照してください。

深層学習の位置づけ

深層学習は、そもそも機械学習における一分野であるニューラルネットワークが発展して形成された分野であり、**図3**のような枠組みとして概観できます。そのため、「機械学習、その

中でもニューラルネットワークを理解したあとに深層学習は取り組むべきものであって、機械学習入門者がいきなり取り組む分野ではない」ことを注意しておきます。

深層学習小史――深層（ディープ）までの道のり

ニューラルネットワークの歴史は古く、1940年代には研究が開始されていた分野であり、最近突如登場した技術ではありません。

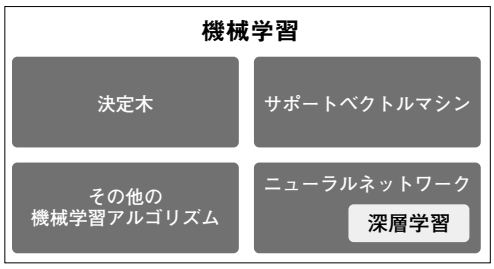
ニューラルネットワークは、脳の一部（ニューロン）の構造を単純化して数学的にモデル化した「形式ニューロン」[注1]から構成されるネットワークです。ニューラルネットワーク研究のおもなブレイクスルーとしては、次が挙げられます（**図4**）。

- 1958年のFrank Rosenblattによるパーセプトロン
- 1986年のDavid E. Rumelhartによるバックプロパゲーションネットワーク
- 2006年のGeoffrey Everest Hintonらによるディープビリーフネットワークなど

これらの発表に伴い、ブームが起きては下火になるということを繰り返しています。とくに2006年ごろからこれまで実現し得なかった多層のニューラルネットワークが実現されたことで、画像認識、音声認識、機械翻訳などさまざまな領域のタスクで、次々と劇的な精度向上が見られるようになります。これを機に「深層学習（Deep Learning）」という分野として確立されます。そして、現在も活発に研究がなされています。

注1）いくつかの入力に対して入力の合計が一定以上になると発火（出力）が起きるしくみ。

▼図3　深層学習の位置づけ



機械学習と深層学習の混同されやすいところ

昨今の深層学習ブームによって深層学習という言葉が独り歩きして、クローズアップされがちです。そこで従来の機械学習技術と比較して、深層学習に対して勘違いされやすい事柄を整理します。

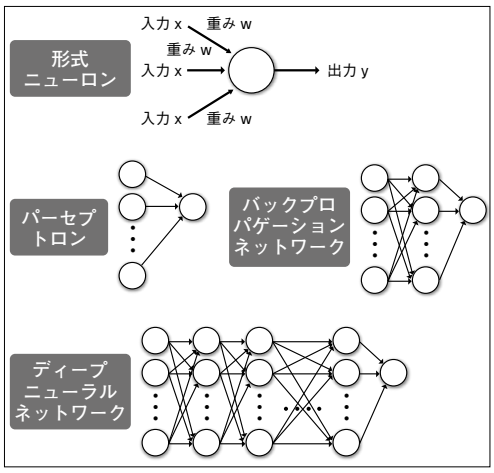
深層学習なら自動で特徴抽出を行える?

画像認識における深層学習の技術として、畳み込みニューラルネットワークという技術があります。この技術により、従来の画像認識技術（局所特徴抽出やコーディング手法など）を要することなく、そのままの画像から学習することが可能となったことで、自動で特徴抽出できるという解説をよく目にするかと思います。

しかしこれが意味するところは、画像データに対する畳み込みニューラルネットワーク技術の特徴であって、あらゆるデータ（社内の文章や、売上などの数値データなど）に対して自動で特徴抽出を行えるということは意味していません。

また、対象が画像であれば何でも自動で都合の良い特徴抽出をしてくれる技術ではありませ

▼図4　ニューラルネットワークの発展



ん。実務で利用するためには、ノイズになるような画像はクロッピング（切り抜き）といった加工をしたりとさまざまな前処理が必要となるため、前処理が不要という意味ではないことに注意しておきます。そのため、「従来の機械学習技術適用時と同様に、前処理を要する」ことが多々あります。

深層学習を適用すれば従来の機械学習と比較して精度が向上する?

深層学習により画像認識や、音声認識、機械翻訳などさまざまな領域で精度向上のブレイクスルーがあったため、深層学習であれば精度向上が期待できそうですが、実際にすべてのタスクで精度向上が期待できるわけではありません。たとえば、マーケティングなどにおける商品の売り上げ予測や、商品のレコメンデーション、広告のクリック予測など、特徴量の作り込みによって精度の良し悪しが大きく決まるような領域では、従来の機械学習技術でも十分な高精度モデルを構築でき、とくに深層学習を必要とするわけではありません。

深層学習では、対象となるタスクの向き不向きやデータの性質が大きく精度に起因します。そのためビジネスにおいて、深層学習を活用することと、従来の機械学習を活用することは大きく異なります。適用したいタスクの性質についてきちんと理解して、深層学習の適用を検討しましょう。

人工知能とは?

よく機械学習と同じように語られるものとして「人工知能」があります。人工知能といえば、SFの世界でも登場するドラえもんやターミネーターのようなロボットのおかげで親しみやすく身近なものとして感じられるため、機械学習と聞くと最初は人工知能のようなものを想像されたかもしれません。しかし、機械学習は人工知能と似ているものの、目的や用途などの面で少し異なります。ここで人工知能とは何か、また機械学習と人工知能の違いを解説します。

汎用人工知能と特化型人工知能

単に人工知能といっても、大きく2種類に分けられます。それは「汎用人工知能」と「特化型人工知能」です。

汎用人工知能というのは、異なる領域にて多様で複雑な課題を解決する知能のことです。汎用人工知能の特徴としては、設計時の想定を超えた新しい問題を解決できる、また自己理解・自律的自己制御が可能であるとされています。このように意識や意欲を持って課題に取り組める人工知能の実現は極めて難しく、現在でも最先端の研究課題となっています。

一方で特化型人工知能というのは、個別の領域において知的に振る舞う人工知能のことです。特化型人工知能は汎用人工知能とは異なり、自律的に学習する能力を持っていないため、人間が課題を発見して人間が能力を高めていく必要があります。特化型人工知能はすでにいくつか実現されています。最近の事例としては、iPhone（iOS）に搭載されているパーソナルアシスタントのSiriや、囲碁の世界トップクラスのプロ棋士を破ったGoogle DeepMind社によるAlphaGoなどが挙げられます。

機械学習と人工知能の混同されやすいところ

普段目に触れている人工知能が特化型人工知能ということがわかったところで、よく混同されがちな機械学習と人工知能（とくに特化型人工知能）の違いを見てみます。さまざまな文献を参照しても諸説あり、明確な定義はなかったのですが、筆者は次のように理解しています。

- 人工知能：人間がするような行動を「人間と同じように」機械にやってもらうこと
- 機械学習：人間が簡単に判断するようなことを「機械がやりやすいように」機械にやってもらうこと

よりはっきりとイメージしてもらうために、人間が利用する例で具体的に見てみます。人間が「機械学習について知りたい」と思ったときに、人工知能と機械学習を道具としてどのように利用するでしょうか。

・人工知能を利用する：iPhoneに向かって「Hey Siri」と声をかけて「機械学習とはなんですか？」と尋ねる。SiriはWebサイトから機械学習のページを検索し、該当しそうなページから「機械学習とは～」を読みあげる
・機械学習を利用する：Google検索で「機械学習」というワードを検索する。Googleの検索エンジンは、膨大なWebページの中から「機械学習」に関連していそうなホームページを列挙し、人間に見てもらう

「機械学習について知りたい」という要望にどちらもコンピュータが対応していますが、さまざま部分で異なることが見て取れると思います。

SiriはiOSが備えている人工知能の機能です。人間が尋ねた「機械学習とはなんですか？」という音声を認識し、その文章を解釈しました。それに対して適切な回答を見いだして、1つに絞り、検索に該当したページの中の該当個所を読みあげています。

一方、機械学習の機能としてのGoogle検索の場合、人間が機械にも解釈しやすいように「機械学習」というワードまで切り出して検索してもらいます。ここで、人間が必要としている機械学習の情報の詳細が何かはわからないため、「機械学習」に関連していそうなホームページを列挙しているだけです。列挙されたホームページから本当に必要な情報は、人間が解釈して見つけなければいけません。

このように、人工知能を利用する場合は「依頼する」形に近く、依頼された人工知能側がどうするかは人工知能自身にゆだねられます。ひるがえって機械学習の場合は「利用する」形に近く、利用のタイミングや利用にあたってのデータの入力方法などは人間自身が判断します。人工知能には部分的に機械学習の技術が含まれますが、このように利用形態においていくつか違いがあることがわかるかと思います。

深層学習と人工知能の混同されやすいところ

これまで解説したように、深層学習は機械学習の一分野のアルゴリズムの総称です。それに対して人工知能はシステムであるため、根本的に両者は異なります。それにもかかわらず、混同して語られやすいのは、人工知能システムに深層学習の機能が組み込まれている場合があるからだと考えています。また、深層学習が人間の脳を模倣して作られた技術という表現が広く伝わってしまったことも、1つの要因であると考えています。そこで混同されやすい点について丁寧に解説します。

深層学習は人間の脳を模倣して作られている技術?

よくニュース記事などで語られるお馴染みの解説ですが、深層学習は人間の脳を模倣して作られている技術という表現を度々目にします。そもそも深層学習のもととなっているニューラルネットワークは、脳の一部（ニューロン）の構造が着想にあるとされています[5]。また、画像認識における畳み込みニューラルネットワークにおいて、畳み込みとプーリングという基本的な画像処理を行いますが、これらは生物の脳の視覚野に関する神経科学の知見をヒントとしています[6]。

このように画像認識における深層学習（畳み込みニューラルネットワーク）については、人間の脳を模倣した技術の着想がいくつか見られるため、このように語られるのだと考えられます。しかしそれを理由に、「深層学習＝人間の脳を模倣した技術＝人工知能」と解釈してはいけません。人間の脳から着想を受けている部分に関しては、画像認識技術に関することがほとんどであって、「画像認識技術＝人工知能」と

いうわけではないからです[注2]。

深層学習だけで人工知能システムを構成しているわけではない

最近、囲碁の世界トップクラスのプロ棋士を破ったGoogle DeepMind社によるAlphaGoについては、深層学習（畳み込みニューラルネットワーク）に加えて、強化学習という学習方法で学習しています。強化学習とは、環境情報（入力）から取るべき行動（出力）を学びます。行動を繰り返し行うことで良い（報酬が高くなる）行動を学んでいくしくみです。

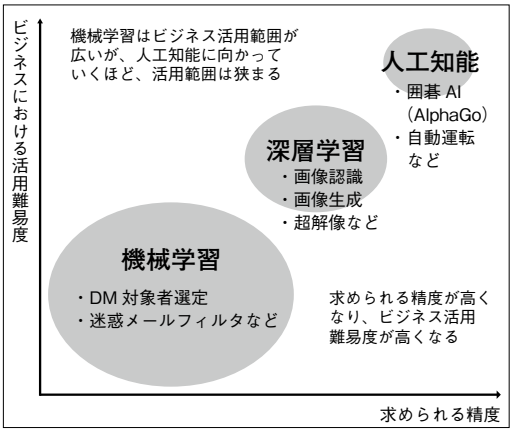
AlphaGoのしくみに関して、大雑把にその役割を分けると、囲碁の盤面を見る「眼」の役割を果たすのが、深層学習（畳み込みニューラルネットワーク）であり、その状態を認識して適切な行動を取るための役割を果たすのが、強化学習です。最近のゲームAIなどには強化学習のしくみが取り込まれており、さまざまなAIにおいて、この「状態の認識」にあたる部分で深層学習が用いられており、深層学習だけで人工知能が構成されているわけではありません[7]。

機械学習、深層学習、人工知能のビジネス活用難易度

これからのビジネスには深層学習や人工知能の活用が必要だと声高に言われるような昨今ですが、機械学習、深層学習、人工知能をビジネスで活用する難易度はそれぞれに異なります（**図5**）。

たとえば、人工知能の例としてのAlphaGoであれば、囲碁のプロ棋士に勝たなければ意味がないですし、人間のように振る舞う必要があるという点で、求められる精度（水準）がとても高いです。一方機械学習は、たとえば、マーケティングなどで活用されるDM送付対象者の選定や、迷惑メールのフィルタリング機能など、多少予測に間違いがあっても、なんとか活用できるものです。

▼図5　機械学習、深層学習、人工知能の活用難易度



このように求められる精度に伴って、ビジネスにおける活用難易度も次第に高くなっていきます。また、機械学習はその多くが予測発見を行うための道具として利用できるため、適用範囲が広いです。一方人工知能では、それぞれの人工知能の機能の特殊性からも、ビジネス活用の範囲は限定されることが考えられます。たとえば、AlphaGoのような人工知能を実現できたところでどうやってマネタイズすれば良いでしょうか？　自動運転が実現できたところでその安全性は保証されるでしょうか？　とても難しい課題です。

このように深層学習ブーム、人工知能ブームの時代ですが、ブームにあやかってそれらを無理に活用しようとせず、機械学習で事足りるかどうか、しっかり検討する必要があるのです。

これからビジネスで活用が期待される深層学習と人工知能

最後に、今後ビジネスで活用が期待されるであろう深層学習と人工知能について解説します。先進的なWeb系企業が実践しているニュース記事から抜粋し、事例を交えて紹介します。

深層学習を使ったサービス

Retty㈱が展開するグルメサイト「Retty」に

注2） ここでの人工知能とは、人工知能システム全体という意味合いで使用しています。

おける、深層学習を使ったコスト削減の事例です[8]。Rettyにおけるユーザが投稿する写真画像を、料理／店舗外観／内観（店内）／メニューの4つに分類する仕事を、深層学習によって自動化しています。従来、この分類の仕事はほかの仕事と併せて外部に発注されており、分類の仕事だけの概算で月に数十万円が掛かっていたそうです。今では、分類の外注費用はほぼゼロとのことです。このように、外注している業務の一部を切り出して深層学習によって自動化することに成功しています。

機械学習により業務の一部を自動化することは、とくに目新しいものでもなんでもなく、これまでも活用されていました。とらえ方によっては、機械学習で簡単に自動化できる範囲が、深層学習によって画像も対象可能になったと言えるでしょう。この例からもわかるように、深層学習を活用すべき対象としては画像データが最適です。そのため多くの企業において、画像データが蓄積され、活用できる状態であることが求められます。

人工知能を使ったサービス

㈱リクルート住まいカンパニーは、スーモカウンターというリアル店舗にて、注文住宅や新築マンションの購入検討を相談できるサービスを展開しています。そのサービスにおいて、WebサイトにAIアドバイザー「スミヨ」という人工知能のチャットボットを導入しています[9]。この「スミヨ」に対してテキストを入力をして話しかけると、リクルートテクノロジーズの研究開発機関であるアドバンスドテクノロジーラボ（ATL）が独自に開発した会話エンジン「TAISHI」によって会話が分析され、会話データベースから最適な回答が選択されるようになっています。

研究開発機関によって開発されているような会話エンジンを、すぐに開発することは非常に難しいでしょう。そのため、人工知能を使ったサービスの導入の敷居はとても高いといえます。また、AIアドバイザーと言っても人間のスタッフが対応するほど細かい要求に対応できるわけではありませんし、すべてをAIアドバイザー任せることはできません。この例からもわかるように、人工知能の導入は非常に難しいうえに、業務効率にも間接的な効果しか見込めず、人工知能ブームの加熱ぶりに対して、人工知能の利活用はとても難しい状況です。

これら状況を理解して、人工知能の活用を検討しましょう。SD

参考文献

[1] Christopher M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer, 2011年, ISBN-13: 978-0387310732.

[2] Kevin P. Murphy, Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series), The MIT Press, 2012年, ISBN-13: 978-0262018029.

[3] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning (Adaptive Computation and Machine Learning series), The MIT Press, 2016年, ISBN-13: 978-0262035613.

[4] 五木田和也, コンピューターで「脳」がつくれるか, 技術評論社, 2016年, ISBN-13: 978-4774184104.

[5] 岡谷貴之, 深層学習（機械学習プロフェッショナルシリーズ）, 講談社, 2015年, ISBN-13: 978-4061529021.

[6] 原田達也, 画像認識（機械学習プロフェッショナルシリーズ）, 講談社, 2017年, ISBN-13: 978-4061529120.

[7] 東京大学 松尾豊, 人工知能の現状と競争政策, <http://www.jftc.go.jp/cprc/conference/index.files/170331data02.pdf>.

[8] グルメサイトRettyのAI舞台裏 - Rettyが50万円で作った儲かるAI：ITpro, <http://itpro.nikkeibp.co.jp/atcl/column/17/032300097/032400001/>.

[9] スーモカウンターWebサイトにAIアドバイザー「スミヨ」登場｜リクルート住まいカンパニー, <https://www.recruit-sumai.co.jp/press/2016/03/webai.html>.

第2章

ディープラーニング入門

CNNで画像分類とドキュメント分類にチャレンジ!

本章では、口コミでおいしいお店を探せるWebサービス「Retty」での機械学習導入の事例をもとに、ディープラーニングによる簡単な画像分類とドキュメントの分類を解説します。ソースコードを入手して、手元で試してみてください。

Author 氏原 淳志（うじはら あつし）　Retty㈱

ディープラーニング（Deep Learning：深層学習）は2012年のILSVRC（ImageNet Large Scale Visual Recognition Challenge）という画像分類コンテストにおいてCNN（Convolutional Neural Network：畳み込みニューラルネットワーク）を用いたチームが圧勝して以降、注目されるようになりました。2015年にはChainer[注1]やTensorFlow[注2]がリリースされ、それまであまり機械学習に手をつけたことがない人も手軽にディープラーニングを試せるようになりました。筆者はまさにこの流行りに乗ってディープラーニングに手をつけた1人です。

筆者がRettyに入社したのは2015年でしたが、当時Retty[注3]での写真の分類は人の手で行われていました。それまで機械学習を手がけたことはありませんでしたが、TensorFlowは初期のころからチュートリアル・サンプルが多く提供されていたので独学しやすく、とくにサンプルとして公開されていたCIFER-10のコードは、少し改造するだけでRettyの写真分類に使えそうだと思えたので自前のMac miniを使って学習させて遊んでいました。それを当社CTOの樽石（本特集第3章を執筆）に見せたところとても乗り気になり、いつの間にか機械学習基盤（後にいうAKIBA。第3章参照）がそろえられ、筆者は機械学習担当としていろいろやらせてもらえるようになりました。

筆者自身もともとは生物系の研究をしていた人間で機械学習や情報系のバックボーンを持っていないので、この2年はサンプルコードや論文を読みながら、画像の分類から、超解像、そしてキャッチコピーの生成や口コミ解析など少しずつRetty内のデータにディープラーニングを応用しています。

本稿ではこの2年で筆者が画像処理から自然言語処理へとディープラーニングを応用していった流れに沿って、とくにCNNに焦点を当ててディープラーニングの手法を解説していきます。

途中載せるコードにはディープラーニング用ライブラリのKeras[注4]を用います。KerasはPythonで書かれていて、バックエンドにTensorFlowかTheano[注5]を使えます。本稿のコードはTensorFlowをバックエンドに使うことを前提にします。

インストールはpipコマンドでできます。

```
$ pip install tensorflow
$ pip install keras
```

注1）URL https://chainer.org/
注2）URL https://www.tensorflow.org/
注3）URL https://retty.me/
注4）URL https://keras.io/
注5）URL http://deeplearning.net/software/theano/

最新のPythonを使っていればpipコマンドはすでにインストールされているはずですが、ない場合は公式サイト[注6]を参考にインストールしてください。

KerasはデフォルトではTensorFlowを使おうとするので、インストールさえできれば設定はとくに必要ないはずです。

どちらのバックエンドを用いても基本的には変わらないのですが、TensorFlowとTheanoで一部互換性がない部分があるので注意してください。

なお、本稿で紹介する画像分類のモデルとドキュメント分類のモデルのソースコードを用意しました。本誌サポートWebサイト[注7]よりダウンロードしていただき、記事を読みながらコードを追ってみてください。

畳み込みニューラルネットワーク

畳み込みニューラルネットワーク（CNN）は画像から特徴を抽出するのにとても優れた性能を誇ります。ディープラーニング以前では、画像の機械学習といえばまず画像から何らかの特徴量を抽出してくることが必要でした。たとえばエッジ抽出であったり、SIFTやSURF[注8]などさまざまな手法があります。しかし、ある問題に対してどのような特徴量を利用していくかを決めるのは専門家の職人芸の部類の話で、素人にはなかなか難しいことでした。

CNNでは、この特徴量の抽出そのものをニューラルネットに任せることができます。画像の特徴量ではなく、画像そのものを入力として使うことができるため、どのような特徴量を利用するかを考える必要がありません。そのうえとても性能がいいので、今や画像を扱う機械学習でもCNNがほぼ一強と言っていいほどよく使われています。

CNNでは畳み込み層とプーリング層を組み合わせて使います。簡単に言うと、畳み込み層は特徴抽出をする層、プーリング層は画像の少々の歪みや移動に対して強くするようにする層です。まずはそれぞれについて説明して、そのあと簡単な画像分類のためのニューラルネットワークを組んでみましょう。

畳み込み層

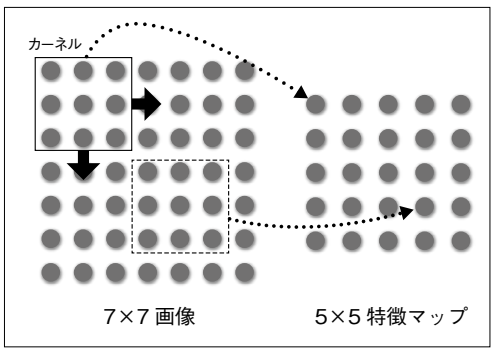
ここでは畳み込み層のしくみを軽く確認しておきましょう。ライブラリを使うときに設定するパラメータが何を意味しているのか、ある程度イメージできればいいです。

畳み込み層のざっくりとしたイメージは、入力をその特徴を表現した特徴マップ（feature map）に変換する、です。

図1は畳み込みで画像を特徴マップに変換しているところを表しています。これは縦7ピクセル×横7ピクセルの画像を縦3×横3のカーネルでストライドを縦1、横1として畳み込みをかけています。さあ、前の文が何を言っているのかわかりませんね。分解して見ていきましょう。

入力に使われているのは縦7×横7の画像です。RGB画像でもいいのですが、単純のためにここではグレースケール画像としましょう。グレースケール画像は1画素が0～255の間の数値をとります。CNNにかけるときは255で

▼図1　畳み込み層の動作イメージ



注6）URL https://pip.pypa.io/en/stable/installing/
注7）URL http://gihyo.jp/magazine/SD/archive/2017/201708/support
注8）SIFT：Scale-Invariant Feature Transform。SURF：Speeded Up Robust Features。いずれも画像の特徴量を抽出する手法。

割って0〜1.0にしたり、128引いてから128で割って-1.0〜1.0にしておくといいでしょう。

次に、縦3×横3のカーネルとは何でしょうか。CNNでは決められたサイズに画像を切り取り、切り取られた画像片に対してカーネル（フィルタと呼ばれることも）で処理して1つの値を変換します。これを画像全体に対して行います。カーネルがどのように画像片を値に変換するのかということが、CNNで学習される部分になります。このカーネルのサイズが縦3×横3だと切り取る画像のサイズは縦3×横3の9ピクセルとなります。

次にストライドとは何でしょう。これは上記カーネルをずらす幅のことです。横のストライドが1であれば、カーネルは横に1ピクセルずつずれながら適用されていきます。縦1、横1なので横にも縦にも1ピクセルずつずれながら画像全体にカーネルをかけていきます。

7×7の画像・3×3のカーネル・縦1、横1のストライドであれば、できあがる特徴マップのサイズは5×5になります（**図1右**）。以上をふまえてKerasでここまでがどのようなコードになるかを見てみましょう。

リスト1はKerasのコードの抜粋です。Conv2Dは画像のような2次元のものに対して畳み込みをかけるときに使います。

第1引数の1はカーネルの数です。カーネルは複数種類学習させることができ、k個のカーネルを使えば出力される特徴マップはk個になります。

第2引数はカーネルのサイズです。横, 縦の順で指定します。

input_shapeは入力される画像のサイズです。(1, 7, 7) の最初の1はチャネル数です。グレースケールなので1ですが、RGBであればR、G、Bそれぞれに値があるので3になります。7, 7は画像のサイズですね。

▼リスト1　Kerasでの畳み込み層

```
from keras.layers import Conv2D

Conv2D(1, (3, 3), strides=(1, 1), input_shape=(1, 7, 7), border_mode='valid', activation="relu")
```

border_modeは現時点であまり気にする必要はないのですが、パディング[注9]のしかたです。畳み込みを普通に行えば、出力される特徴マップのサイズは入力される画像のサイズより小さくなります。そこで画像の周りに0を付け足して7×7画像を9×9画像にしてやると、出力される特徴マップのサイズが7×7になり、入力画像のサイズと同じになります（**図2**）。こうしたほうが都合がいい場合もあるのですが、今回はパディングは行わないのでそういうものもあるんだと思っておけばいいです。

activationは活性化関数を指定するもので、これも現時点ではあまり気にする必要はないです。ニューラルネットの層を重ねた際の表現力を向上させるものだと思っておけばいいでしょう。分類問題の最終的な出力にsoftmaxという活性化関数がよく使われるので、この名前は覚えておくといいでしょう。

どうでしょうか？　イメージはつかめたでしょうか。要はあるピクセルとその周りの一定範囲にあるピクセルをまとめて特徴値を計算するというのを、全ピクセルに対して行っているという感じです。

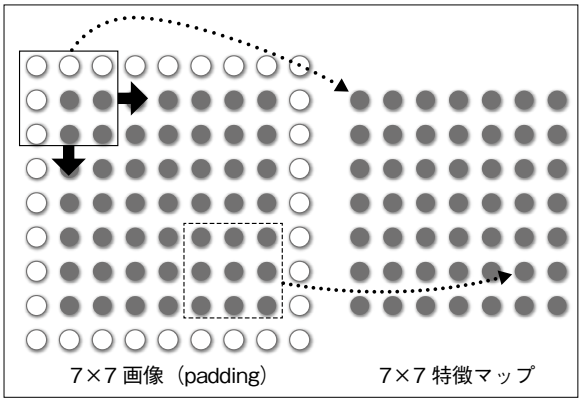
プーリング層

ここでは畳み込み層とセットで使われるプーリング層について説明します。ただ、プーリング層はいろいろな種類があって必ずこうなっているというものでもありません。プーリング層の役割と、代表的なプーリング層がどうやってその役割を果たしているのかを見ていきましょう。

前にプーリング層は、画像の少々の歪みや移動に対して強くする層、と言いました。たとえば猫が走っているところを、固定したiPhoneでバースト撮影（高速連写）した場合を考えて

注9） padding。簡単に言うと、足りない部分を埋めること。

▼図2　padding

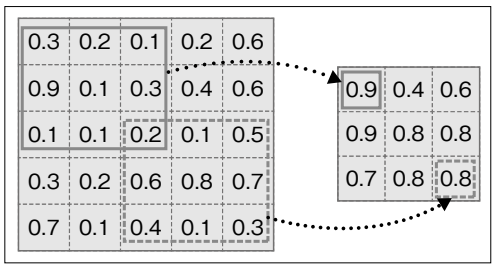


みてください。連続したバースト写真にはほんの少しずつ猫がずれて写っているでしょう。これをそれぞれ畳み込み層に入力した場合、同じ猫を同じ場所で撮影しているにもかかわらず、特徴マップは変わってしまいます。もちろん少しずれているんだから特徴も変わると言えばそのとおりなのですが、たとえば猫と犬の画像を分類する分類機を作ろうとした場合には、そこまで細かいずれなどは必要ない情報なわけです。

そこでプーリング層では特徴マップのある一定の範囲について情報をまとめてしまいます。こうすることでずれや歪みを吸収して結果が安定するようにするのがプーリング層の役目です。

たとえば、Max Poolingというプーリング層は決められた範囲の中の最大の値を特徴値として採用します。先ほど7×7の画像から5×5の特徴マップを取り出しました。**図3**ではその5×5の特徴マップの特徴マップに対して、3×3のサイズでMax Poolingをかけています。

▼図3　Max Poolingの動作イメージ



Kerasで書けば次のようになります。

```
from keras.layers import MaxPooling2D

MaxPooling2D(pool_size=(3, 3))
```

プーリング層にはほかにも、最大ではなく平均を採るAverage Poolingや、全体の最大、4分割した画像のそれぞれの最大、16分割した画像のそれぞれの最大……というように画像を格子状に分割していき、それぞれの最大を採っていくSpatial Pyramid Poolingのような特殊なものも存在します。

プーリングをかけると特徴マップから位置の情報がなくなるので、超解像のように画像生成をやる場合などでは、畳み込み層のみを使ってプーリング層は使わない場合もあります。

CNNでの画像分類

画像分類を行う場合は上記の畳み込み層とプーリング層を、畳み込み層→プーリング層→畳み込み層→プーリング層→……と多層に重ねるのが普通です。最初の畳み込み層・プーリング層は画像から特徴を抽出していますが、その次では低次の特徴から高次の特徴を抽出していると見ることができます。

たとえば、人の顔の画像からまずはエッジが抽出されて、次に輪郭、さらには鼻や目・口というふうに少しずつ複雑な特徴をとらえていっているという感じです。以前Googleが、YouTubeの動画から画像を切り出して学習をさせたところ猫を認識する層ができて、それを可視化すると猫の顔に見えるものになったという話題がありましたが[注10]、これなどはおもしろい例かもしれません。

そうして抽出された特徴マップを全結合層に渡します。全結合層も複数層重ねるのが一般的で、最終的に分類したいクラス数の次元まで次

注10）URL https://googleblog.blogspot.jp/2012/06/using-large-scale-brain-simulations-for.html

元を削減します（**図4**）。

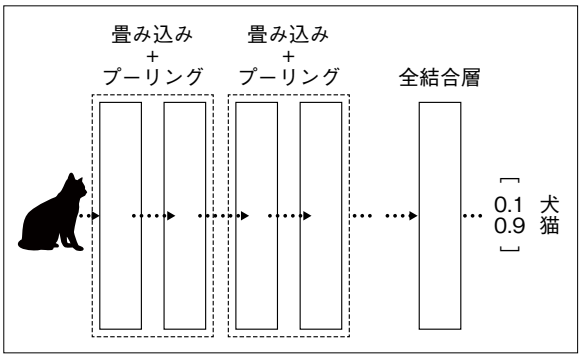
本誌サポートWebサイトに用意したKerasのモデル「image_classify.py」は、32 × 32のRGB画像を2つのクラスに分類するモデルです。実行するには画像データを自分で用意する必要があります。**図5**のようなディレクトリ構造にして、画像を入れておいてください。簡単なものならこれでも分類できるでしょうが、さらに精度を上げていくならCNNの層を増やしたり、Batch Normalization層の追加、Dropout層の追加、パラメータの調整などいろいろ工夫ができます。ぜひいろいろ試してみてください。

CNNの自然言語処理への応用

ディープラーニングにおいて自然言語処理にはよくRNN（Recurrent Neural Network：リカレントニューラルネットワーク）が使われています。RNNはシーケンシャルな入力を扱うのが得意なためです。しかし一般的に、RNNはCNNに比べて学習が遅いです。そこで自然言語処理にもCNNを使った論文が去年あたりからよく出るようになりました。たとえば最近では、Facebookが機械翻訳をCNNで実装したという話題が5月に出ました[注11]。ここでは自然言語処理の中でも、とくに文章の分類にCNNを使う方法について紹介します。

注11）URL https://code.facebook.com/posts/1978007565818999/a-novel-approach-to-neural-machine-translation/

▼図4　CNNでの画像分類の動作イメージ



自然言語の前処理

自然言語をディープラーニングで扱うにはまず必要なことがあります。それは文章をニューラルネットワークに流せる形に変換してやることです。つまりは文章を行列で表現することです。これにはどんな方法があるでしょうか？

有名なのはword2vecでしょう。word2vecは単語をベクトルとして表現する方法です。文章は単語の配列なので、単語がベクトルとして表現できれば文章はベクトルの配列として表現できます。しかし、文章を単語の配列にするというところで、日本語の自然言語処理で考えなくてはいけない問題に突き当たります。

ある文を単語の配列にするということを考えたとき、英文ではとても簡単です。なぜなら英文では基本的に単語は空白で区切られているためです。しかし、日本語では単語の切れ目は空白区切りのような単純なルールでは区切ることができません。

▼図5　image_classify.pyの実行に必要な画像データの配置（犬と猫の画像分類の場合）

```
……
+ー image_classify.py
|
+ー data
      +ー test（テストデータ）
      |      +ー cat
      |      |      +ー ○○.png（猫の画像）
      |      |      |
      |      |      ……
      |      +ー dog
      |             +ー ○○.png（犬の画像）
      |             |
      |             ……
      |
      +ー train（学習データ）
            +ー cat
            |      +ー ○○.png（猫の画像）
            |      |
            |      ……
            +ー dog
                   +ー ○○.png（犬の画像）
                   |
                   ……
```

```
How I want a drink, alcoholic of course, ↵
after the heavy chapters involving ↵
quantum mechanics.

すももももももものうち。
```

日本語の文章を単語に分解するために用いられるのが形態素解析です。形態素解析用のツールにはMeCabなどがあり、これを使えば文を単語に分解できます。

```
すももももももものうち

すもも 名詞,一般,……
も 助詞,係助詞,……
もも　名詞,一般,……
も 助詞,係助詞,……
もも　名詞,一般,……
の 助詞,連体化,……
うち　名詞,非自立,副詞可能,……
```

しかし、この形態素解析には弱点があります。それは未知の単語は扱えないということです（**図6**）。もちろん辞書に単語を追加していけばいいのですが、それが困難な場合もあります。

Rettyの場合、メニュー名や店名などの固有名詞や口コミならではの口語表現などが数多く、辞書に追加しても新しい単語は次々に出てくるため、形態素解析できちんと文章を分解できるようにするには辞書を整備し続ける必要がありました。もちろんRettyでは自然言語処理の基盤を作るためそのような辞書も整備しているのですが、これはとても手間のかかる仕事です。

ここで前に説明した画像の分類について思い出していただきたいのですが、画像では写っている物体をいちいち分解してCNNに渡したりはしていませんでした。画像はピクセル単位でCNNに入力されていました。ならば自然言語でも同じことができるのではないでしょうか。つまり、文を単語ではなく文字に分解して1文字単位で処理する。そして単語というような構造はニューラルネットワークに見つけさせる。そういうことができるのではないか、ということです。それがCharacter-level CNNです。

▼図6　形態素解析がうまく行かない例

```
# 辞書にない固有名詞
いぶりがっこのクリームチーズのせ

いぶり 動詞,自立,……
がっ　動詞,接尾,……
この　連体詞,……
クリームチーズ 名詞,一般,……
のせ　動詞,自立,……

# 辞書にない口語表現
ぉレ£∋ぅ⊇〃±〃レヽｍａ£

ぉ 名詞,一般,……
レ 名詞,一般,……
£∋ 名詞,サ変接続,……
ぅ 名詞,一般,……
⊇〃±〃 名詞,サ変接続,……
レ 名詞,一般,……
ヽ 記号,一般,……
ｍａ 名詞,固有名詞,組織,……
£ 名詞,サ変接続,……

※「おはようございます」らしい
```

Character-level CNN

Character-level CNNは2015年ごろから見るようになった手法です[注12]。Character-level CNNのありがたい特徴の1つは、上で述べたとおり形態素解析が必要ないことです。文を文字単位でCNNに入力して特徴マップに落とし込み、全結合層に流し込んで分類を学習します。具体的にその手法を見ていきましょう。

まずは文を文字単位に分解して文字の配列にします。

注12）URL https://arxiv.org/abs/1509.01626

```
いぶりがっこのクリームチーズのせ
  ↓
[い, ぶ, り, が, っ, こ, の, ク, リ, ー, ム, チ, ー, ズ, の, せ]
```

次に個々の文字に固有のIDを割り当てます。自分でIDを振ってもいいのですが、ここは簡便のためUNICODEの値を代用しましょう。

```
[い, ぶ, り, が, っ, こ, の, ク, リ, ー, ム, チ, ー, ズ, の, せ]
  ↓
[12356, 12406, 12426, 12364, 12387, 12371, 12398, 12463, 12522, 12540, 12512, 12481, 12540, 12474, 12398, 12379]
```

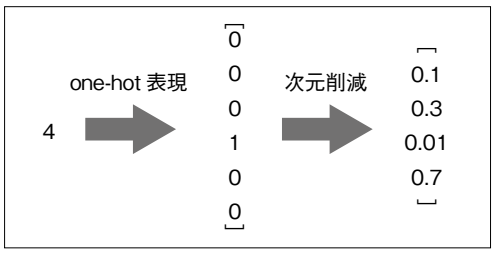
CNNで扱うには固定長であったほうが都合がいいので、扱う長さを決めて、その長さに足りないなら0でパディング、長すぎるなら決めた長さで打ち切ります。

```
[12356, 12406, 12426, 12364, 12387, 12371, 12398, 12463, 12522, 12540, 12512, 12481, 12540, 12474, 12398, 12379]

10文字までなら10文字までで打ち切り
→ [12356, 12406, 12426, 12364, 12387, 12371, 12398, 12463, 12522, 12540]

20文字までなら足りない分0で埋める
→ [12356, 12406, 12426, 12364, 12387, 12371, 12398, 12463, 12522, 12540, 12512, 12481, 12540, 12474, 12398, 12379, 0, 0, 0, 0 ]
```

▼図7　IDの埋め込み表現



IDを埋め込み表現に変換します。これはIDをone-hotベクトルにして、その次元削減を行うところです（**図7**）。

one-hotベクトルとは1つの要素の値だけ1で、それ以外が0のベクトルです。この場合、扱うIDの最大値と同じ大きさの次元のベクトルを用意し、IDの値が示す要素だけを1にしてそれ以外を0にします。つまり、文字種が0xffff個あるのであれば0xffff次元の0ベクトルを用意し、「あ」が12355番めの要素なら12355番めを1にします。これをたとえば128次元に次元を削減することで、1文字が128次元のベクトルとして得られます。

次元削減というと面倒そうですが、幸いにも埋め込み表現の学習はたいていのライブラリでAPIとして用意されているはずです。Kerasでは「Embedding」というのがそれで、次のように書きます。

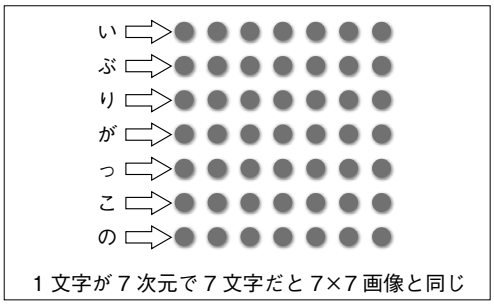
```
from keras.layers import Embedding

Embedding(0xffff, 128)
```

第1引数にone-hotベクトルの次元、第2引数に削減後の次元を指定するだけでお手軽です。

さて、ここまでの処理で実はすでに文字列が画像と同じように扱える状態になっています。文を固定長配列にするときに7文字にしたとしましょう。そして埋め込み表現で文字を7次元のベクトルにしたとしましょう。そうすると、

▼図8　文字列が画像と同じ形になっている



これは実は7×7のグレースケール画像と同じデータの形になっています（**図8**）。

文字列が画像と同じ形になったので、あとはこのまま畳み込みを行うことができます。

ただ、ここで少し工夫をします。畳み込みではカーネルのサイズを決めて、それをスライドさせながら特徴量を計算していました。ここでカーネルのサイズを、横を1文字の次元サイズ分、縦を2文字分にして畳み込みを行います。そうすると、文字列から2文字分ずつ取りながら特徴マップを計算する畳み込みができます。カーネルサイズを縦に3文字分、4文字分……と変えていけば、それぞれ3文字分、4文字分……ずつの特徴マップができあがります（**図9**）。

これは何をしているのかというと、n-gram[注13]を畳み込みで再現しています。そうして複数のカーネルで畳み込みをした結果をそれぞれプーリング層に流して、その後1つに結合することで複数種類のn-gramをした結果の特徴マップを作成します（**図10**）。

こうして得られた結果を画像分類のときと同じように全結合層に流せば、自然言語の分類モデルができあがります。実際のコードは本誌サポートWebサイトに用意した「char_cnn.py」です。

Character-level CNNの応用例

筆者がRettyで実際にCharacter-level CNNを応用した事例について紹介します。

Rettyではお店に目的が割り振られています。これによってたとえば「デート」という目的に合った店を探すということができます。ここで「デート」の目的に割り振られている店の口コミを「デート目的店口コミ」とします。そしてそれ以外の店の口コミを「デート以外目的店口コミ」とします。これらを教師として、口コミを「デート目的店口コミ」か「デート以外目的店口コミ」に分類する分類器を作成しました。

この分類器を使うと、たとえばある店の口コミをすべて分類器にかけたとき、9割の口コミが「デート目的店口コミ」だった場合、その店はデートに使える店なんじゃないかという判断ができるのではないかと考えました。実際の例を見てみると、

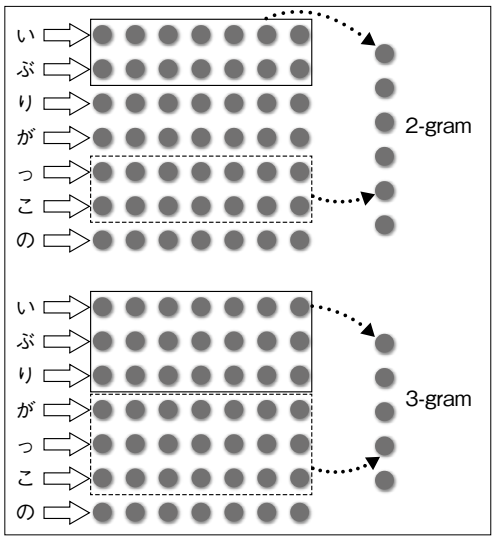
> 武蔵小山の焼き鳥ワインのお店。ここは全般美味かった!値段は決して安くは無いですが、ワインも全てBioワインとの事。焼き鳥はおまかせコースがお勧めとの事でしたので、そちらで。店員さんの接客も最高で是非行ってみて下さい。

この口コミは「デート目的店口コミ」率が99.99%となりました。しかし同じく焼き鳥の店ではあるものの、次の口コミは0.001%以下でした。

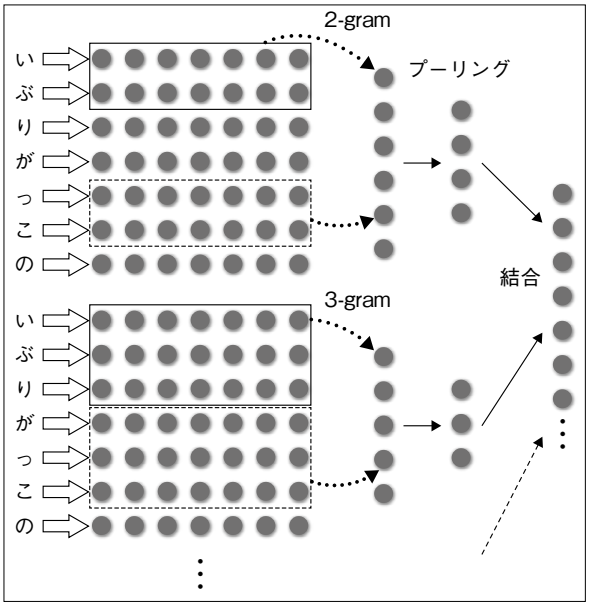
> 駅直結‼という立地の良さから選びました?焼き鳥、水炊きなどなど頼みましたが、鳥がぷるぷるで美味しかったですよ‼仕事終わりの方たちがサクッと一杯…というイメージですかね。でも値段もリーズナブルでなかなか良かったです♪

注13）全文検索などで使われる手法で「N文字インデックス法」「Nグラム法」などともいう。隣り合うn個の文字をセットとして扱う方法。たとえば「いぶりがっこ」を2-gramにすると「いぶ」「ぶり」「りが」「がっ」「っこ」、3-gramなら「いぶり」「ぶりが」「りがっ」「がっこ」。

▼図9　CNNでn-gramを真似する



▼図10　複数の畳み込み結果の結合



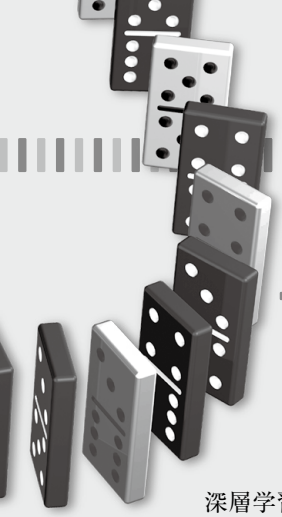
このように同じ焼き鳥でもきっちりとデートに使えそうな店の口コミを見分けてくれるようになりました。この分類器は「デート」にアサインされている店が問題ないか、「デート」にアサインされていない店で「デート」にアサインしたほうがいい店はないか、などの調査に利用されています。

Character-level CNNのさらなる応用例

ここまで、Character-level CNNで文章の分類を行う例を紹介しました。分類ができるようになったということは、文から特徴を抽出できたということです。全結合層を最後まで使わず、たとえば128次元くらいにまでした層から別のニューラルネットワークにつなぎこむことで、文を128次元の特徴ベクトルに変えてその特徴を利用するネットワークを構築することができます。文の特徴を得ることができるなら、たとえばある店の口コミ全体の特徴を抽出して、その結果をLSTM（Long short-term memory）に流してキャッチコピーを自動生成する、ということができるかもしれません。実はこれはすでに挑戦していて、ある程度はそれらしいキャッチコピーを作ることはできています（実用には回せないレベルでしたので要改善ですが……）。画像でもCNNで得られた特徴を使ってさまざまなことが試されていますので、Character-level CNNを活かせばさらにおもしろいことができるかもしれません。

最後に

本稿ではCNNを画像分類から始め、自然言語の分類へと応用していった流れについて紹介しました。イメージさえつかめれば、ライブラリを使って自分独自のニューラルネットワークを構築することもできるでしょう。CNNは、画像系だとほかにも超解像やGAN（Generative Adversarial Network）での画像生成、写真のスタイル変換など広範囲で利用されており、自然言語処理でも機械翻訳に利用されるなど非常に応用範囲の広いものですので、ぜひいろいろ挑戦してみてください。SD



COLUMN 1

自然言語処理エンジニアから見た深層学習

Author 竹野 峻輔（たけの しゅんすけ） Retty㈱

深層学習（ディープラーニング）がもたらした変化は、多分にもれず自然言語処理においても大きな変化をもたらしました。この変化によって、第2章で紹介した文字単位で機械学習モデルを構築するような、深層学習以前の自然言語処理からすれば力技のような手法も容易に実現できるようになりました。本コラムでは、これまでの自然言語処理の処理方式との比較を行い、あらためて深層学習によってもたらされた変化について見ていきます。

テキスト分類から見る深層学習による変化

第2章と同様テキスト分類の問題を対象に、典型的な処理を**図1**に示します。

図中に示したような主題とする問題（今回の場合はテキスト分類）を、いくつかの部分問題に切り分けて取り組むことが多いです。今回の例では「前処理」「構造解析」「特徴量抽出」「モデルの構築」「評価」の5段階に切り分けています。このような、上流の処理の結果を受けて下流の処理の入力へ出力をつなげていく処理方式を逐次処理（パイプライン処理）と呼びます。

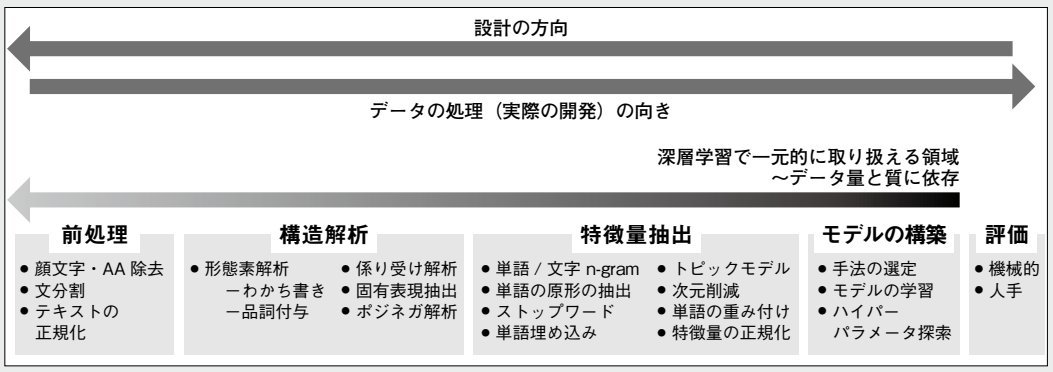
実際の機械学習のプロジェクトに取り組む際には下流の設計から行い、下流の処理が十分に機能するような上流の処理を設計します。各処理の詳細を下流から追って見ていきましょう。

評価

機械学習のモデルの質を評価するものです。学習用のデータと評価用のデータを分けて評価することで、学習時に含まれていないデータに対し性能が維持できるかを確認します。評価といっても観点はさまざまです。2値分類ならば（予測が合っている数）／（予測した数）といった正確さ（accuracy）の評価になります。また機械的な評価だけでは「モデルの間違え方の妥当性」や「間違えてほしくない問題」といったような質的観点からの評価は難しいため、人手による評価も考えられます。

これらの評価の設計は、モデルをどのように構築するか？を決める重要な指針となります。

▼図1　典型的な自然言語処理の流れと深層学習の対比



モデルの構築

実データからモデルの構築を行います。有名なモデルとして、たとえばロジスティック回帰（Logestic Regression）やランダムフォレスト（Random Forest）、サポートベクターマシン（Support Vector Machine）といった手法が挙げられます。ハイパーパラメータと呼ばれる手法ごとに調整可能な値を、評価用のデータセットとは別の開発用データセットを利用し、決定する必要があります。

この部分に関しては、入力と出力を抽象化できるため、共通化も行いやすく、ユーザはほとんどの手間なく記述できます。

特徴量抽出

下流のモデルの構築においては、テキストデータのままでは取り扱いづらいため、計算機が取り扱いやすいベクトルまたは行列表現になおす必要があります。自然言語処理では単語ごとの頻度を数えただけのBag-of-Words（BoW）表現と呼ばれる形式がよく利用されます。これだけでは単語ごとに隣接した単語の情報が失われてしまうため、連続した単語をひとかたまりの単語と考えて取り出すn-gram（第2章参照）もよく使われます。

この変換の際に後段のモデルが学習しやすいようなさまざまな工夫を凝らすことで、精度の大幅な改善が期待されます。このことから特徴量抽出もしくは素性（そせい）抽出と呼ばれています。たとえば、LDA（Latent Dirichlet Allocation）を始めとするトピックモデルや、word2vecの単語埋め込み（分散表現）、TF-IDF（Term Frequency-Inverse Document Frequency）などによる単語の重みづけ、ICA（Independent Component Analysis）やPCA（Principal Component Analysis）やNMF（Non-negative Matrix Factorization）など効果的に学習を行うためのさまざまな工夫がこれまでに考案されています。

構造解析

自然言語処理の特有の処理となるのがこの構造解析です。抽出したい特徴量に合わせて、テキストに構造情報を付与します。日本語でいえば、英語のように単語間の区切りが明確ではありません。英語のように空白区切りを付与する処理を分かち書きと呼びます。品詞付与も併せて行う場合には形態素解析と呼びます。この処理ののち、係り受け解析や固有表現抽出、文章がポジティブな意味合いを持つのか、ネガティブな意味合いを持つのかを解析するポジネガ解析など、モデルの学習に必要な特徴量に合わせて解析を行います。

前処理

アスキーアートや顔文字などが含まれるテキストに対しては、意図しない解析結果を生み出し、下流の処理のノイズになり得ます。これだけでなく、テキストを1行1文と整えるための変換を施したり、テキスト中に存在する制御文字も取り除いたりする必要があります。これらを正規表現などを駆使し、あらかじめ取り除くことで下流の処理を安定させることができます。

深層学習は銀の弾丸になり得るか？

深層学習によってもたらされた変化は非常に大きいものです。ここまで挙げた前処理、構造解析、特徴量抽出、モデルの構築を行わずに、深層学習だけでの実現も可能になりました。逐次処理と対比して一気通貫（end-to-end）処理と呼ばれることも多いです。この変化は非常に驚異的なことです。

自然言語処理を始めたい、と考えたときには**図1**に示したような処理を上流から構築する必要があります。応用的な問題であればあるほど、この「ヤクの毛刈り」のように連なる処理を目的のタスクごとに作る必要がありました。また改善を施すためには形態素解析をはじめとする

構造解析について理解する必要があるため、前提となる知識は多く、初期参入は大変だったように思います。

それに比べて、自然言語処理だけでなく画像処理や音声処理などにも共通して使える枠組みである深層学習は「まずはやってみる」という観点から初心者にとってはむしろ簡単といえるのではないでしょうか。

深層学習に置き換えるトレードオフ

深層学習は非常に柔軟な枠組みの1つではありますが、一方でその能力の限界についても同様に把握する必要があるでしょう。たとえば第2章で解説したCNNを利用した文書分類のモデルでは特徴量抽出、構造解析、前処理を省略したモデルの学習を行うことができます。しかしながら、十分な精度を担保するためには比較的大量のデータと学習時間を必要とします。これは、CNNを利用したテキスト分類では、入力をただの文字のつらなり（＝画素）として捉えるため、意味的なまとまりをモデルに獲得させるために多くのデータを要するためです。

誤りの原因を分析し、改善することを考える点においても深層学習は不利になりやすいです。深層学習のモデルの中身は階層的かつ膨大な数値の羅列であるため、出力の決定過程が人間にとって不透明であるためです。「良い結果は出るが、なぜ出るかわからない」といった状況に陥りやすくなります。

すべてを深層学習に置き換えてしまうことで「人の手が介在しやすい」部分をあえて消しているという点について理解はしておく必要があります。

実問題に対し向き合うために

ここまで、自然言語処理における機械学習について紹介を行ってきました。このコラムの最後として、少し視野を広げ、自然言語処理に限らず、機械学習を利用したプロジェクトに対して取り組む際に必要な視点について考えてみましょう。

根本的に重要になるのは、データとモデルと問題設定の3つの観点をすべて持つことです。この3つの関係性を**図2**に示します。

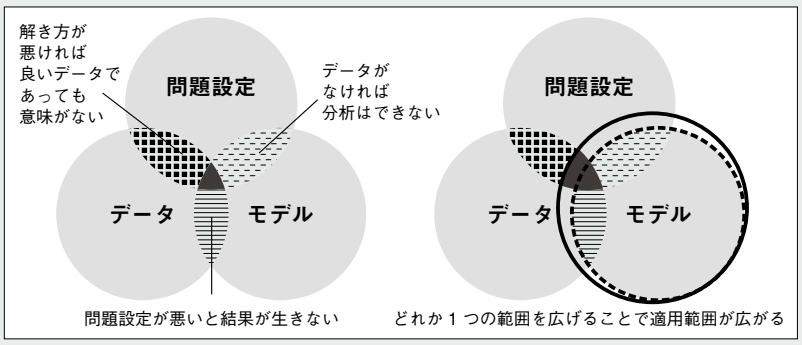
データにおける観点とは、実際に利用できるデータの量や性質、どの程度そのデータが整備または保守されているのかといった観点もここに含まれます。

問題設定に対する観点とは、解きたい問題の種類に加えて、プロジェクトに対する実際の成果の要求の質や問題に取り組める時間、そして、実際に運用するうえでの制約となる観点です。

モデルの観点とは、その問題をどのように定式化し、どのように解くかに対する観点です。大量のデータ量を必要とする深層学習を選ぶか、比較的少量のデータセットにおいても性能を発揮できる手法を選ぶか、といった手法選択の部分となります。

成功する機械学習のプロジェクトの条件は、これら3つの観点のバランスをうまく取ることができたプロジェクトと言えます（**図2**の黒色

▼図2　機械学習を利用したプロジェクトに対して重要になる3つの観点



で塗りつぶした部分)。逆にこれら3つのうち1つでも疎かにすることで、そのプロジェクトが失敗する可能性は途端に高くなります。これを**図2**を使って説明します。

良いモデルと良い問題設定であったとしても、学習・評価するための適したデータがなければ、当たり前ですが機械学習やその分析すら行うことはできません。問題設定に対して関係性のないデータから正しい分析・予測は行えません。

また良いデータと良いモデルが用意できたとしても、問題設定が厳しいものであれば実際に運用していくことは難しいでしょう。とくに決定的になりやすいのが、機械学習のモデルが出力する誤りに対してどれぐらい許容できるかです。誤りがほとんど許されないような問題設定であればあるほど、機械学習を活用するのは難しいといえるでしょう。

そして良い問題設定と良いデータがあったとしても、それに適うモデルを用意できなければ実際に運用することはできません。500ミリ秒以内に出力を返す必要があるにもかかわらず、出力に1秒以上かかってしまうような場合は、いくら精緻(せいち)で高精度なモデルが用意できたとしても運用することはできないでしょう。

機械学習プロジェクトとは、これら3つの観点の折り合いを探し続けることにほかなりません。3つの観点を同時に満たすというのは一筋縄ではいかないのですが、一方で、これらの3つの観点は、互いに互いを補うことができるようになっています。

問題設定が難しい場合には、複数のデータを組み合わせる、もしくは複数の手法を組み合わせることで問題設定の難しさに対応することを考えてみましょう。一方で、データが少ない場合には、より少ないデータで動く手法の検討や、問題設定を単純にすることを考えるべきです。モデルの能力に限界がある場合には、異なるデータセットを組み合わせる、もしくは問題設定をもっと単純に変え、モデルの能力が発揮できやすくすると良いでしょう。

この中で個人単位でなんとかできる観点、言い換えると、属人的な部分はモデルの観点でしょう。さまざまな手法の向き・不向き、組み合わせ方を知ることで、より広い問題設定に対して取り組むことができるようになります。

他方で データや問題設定については、個人単位というよりも、対象とするサービスや組織など個人よりも広い部分に依存する部分です。機械学習を活用できる幅を増やそうと考えた場合には、これら2つの観点に対する整備を組織的に作っていく必要があります。

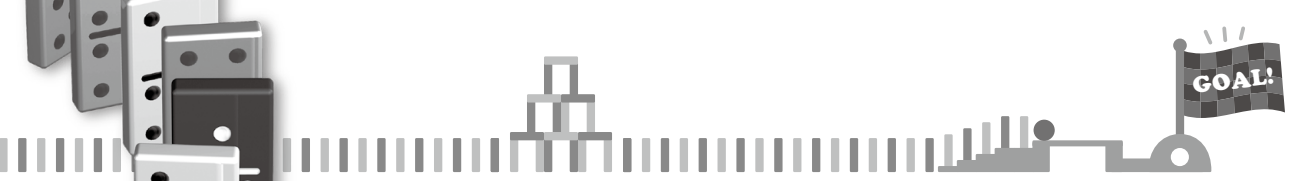
「ビッグデータ」というキーワードを走りに近年では「人工知能」というキーワードのもと、機械学習やデータ分析に大きな注目が集まるようになりました。現実の問題の多くは、データの複雑性が人間の理解の範疇(はんちゅう)を超えるような問題ばかりで、このような問題に対して深層学習をはじめとする機械学習は、万能ではないながらも、これらをうまく取り扱える現状唯一の手段といっても過言ではありません。これをうまく活用することで、より良いサービス作りが進むことには違いありません。

一方で、うまく利用するためには機械学習の手法に対する理解だけでなく、活用するための頑健なログ基盤を整備したり、機械が扱いやすい形式にデータを整理したりするなど、周辺の整備を行っていくことも大事であることは忘れてはなりません。より良いしくみ作りを考えていきましょう。

終わりに

本稿では、前半においては自然言語処理におけるこれまでの典型的な処理の流れと、深層学習を導入することでの利点や欠点について説明を行いました。後半では、少し抽象度高く、機械学習をいかにうまく活用するかについて、話の焦点を合わせた内容となりました。本稿をご覧になったみなさんの参考になれば幸いです。

SD



第3章

低予算ではじめる

機械学習用自作マシンのポイント

本章では、前章で解説した「Character-level CNN」をGPUを使って高速に処理する、自宅で使えるコンピュータの作り方を紹介します。すでに自宅にデスクトップコンピュータがあることを想定していますが、予算は約2万円です。

Author 樽石 将人（たるいし まさと） Retty㈱

GPUとは？

GPUはGraphical Processing Unitの略で、もともとコンピュータグラフィックス（CG）処理を高速に行うための専用機器として開発されました。CGは、ポリゴン処理やテクスチャマッピングといった処理を行いますが、これらの処理は、浮動小数点演算などの数学的処理を一度にたくさん行うため、たくさんの計算コアが必要です。CPUは複雑な命令を駆使して、さまざまなことを汎用的に行うことができますが、反面、1つのCPUコアのサイズが大きいため、CPUのコア数を増やすのは容易ではありません。またさまざまなことを処理できるようにするために、トランジスタの数が増えるため、消費電力も増えてしまいます。

GPUは、さまざまなことを汎用的に処理する部分を省き、CG処理に必要な数学的計算処理に絞ることで、1つのGPUコアのサイズを小さくし、代わりに非常に多くのコアを搭載したデバイスです[注1]。このたくさんのGPUコアを利用することで、ポリゴン処理やテクスチャマッピング処理を高速に実行可能になりました。また、トランジスタの数も減らせるため消費電力も少なくて済みます。GPUは電気をたくさん使うという話を聞くこともあると思いますが、実はGPUは省電力な計算ユニットです。省電力のための技術発展も著しく、その結果自宅で動かすことも十分可能です。

このようにGPUは、もともとCG処理のための専用機器として開発されてきましたが、数学的な計算処理を大量に行う別の用途でも活用が可能です。これらの試み・技術をGPGPU（General-Purpose computation on Graphics Processing Units）といい、その代表的な活用例が機械学習です。

GPUの歴史

CG処理とGPUの誕生

CG処理を行うための機器はコンピュータ黎明期から開発されてきましたが、とくに1990年代にグラフィックチップの開発で大きく発展し、さまざまな企業がその技術開発にしのぎを削ってきました。一時は約70もの企業がグラフィックチップの開発を行っていたほどです。その後、1999年に米NVIDIA社が「1秒で1,000万のポリゴンを処理でき座標変換などができるシングルチッププロセッサ」GeForce 256を、

注1）たとえばNVIDIA製のエンタープライズ向けGPU Tesla P100は、3,584個のコアを搭載しています。

GPUとして発売したことで、GPUという呼び名が一般化し、浸透していきました。

GPGPUへの応用とCUDA

GPUのさらなる技術発展と、科学者や研究者が複雑な演算を要する課題に取り組むための道具へのニーズから、2006年にNVIDIAがCUDAという「GPUのグラフィック処理機能を汎用的な数学計算用途に活用できる技術」を発表しました。CUDAはNVIDIA社のGPUのみで利用可能なGPGPUアーキテクチャですが、現在、機械学習を行ううえでは世界的に一般的な基礎技術となっています。なお、そのほかのGPGPUアーキテクチャとしては、AMD社のAMD Stream、マルチベンダにおける互換性の試みであるOpenCLなどがあります。

CUDAコアを搭載したGPUの種類

CUDAには新しい技術が次々と投入されているため、GPUごとに使える機能が異なります。そのため、CUDAにはCompute Capabilityというバージョン番号が付いていて、利用する機械学習フレームワークの対応バージョンをもとに、適切なGPUを選ぶ必要があります。たとえば、機械学習を行うのに一般的なTensorFlowを使うにはバージョン3.0以上が必要です。ただし、最近は組み込み向けなどの小さなGPU以外は十分なCompute Capabilityを確保しているのであまり気にしないでもよいでしょう。

各GPUの対応Compute Capabilityは、NVIDIAのWebサイト[注2]を参照してください。

主要なNVIDIA GPU製品ラインナップ

NVIDIAは、GPUの利用用途ごとにさまざまな製品ラインナップを持っています。ここでは主要なものを紹介します。

GeForce

コンシューマ向けのGPUです。NVIDIA自身はGPUを販売しておらず、各メーカーがCUDAコアをNVIDIAから調達して、製造・販売しています。おもにPCゲーム用のGPUとして販売されています。ただし、GPGPUとしても利用可能でCompute Capabilityも最新のため、自宅でGPUを使った機械学習を行うには最適な製品です。本特集でもこれを使います。

NVIDIA Quadro

NVIDIA自身が製造・販売しているGPUです。おもにハイエンドなCG制作が必要な開発者向けの製品です。NVLinkというNVIDIA独自のデータ転送バスを搭載することで、複数のGPUを使ったCG処理のスピードを大幅に向上します。映画制作などの現場で大活躍します。

また、機種によっては半精度（FP16）の浮動小数点演算に対応しているものもあり、単精度に比べ大幅な高速演算を行うことも可能です。

NVIDIA Tesla

Teslaはサーバ向けGPUです。クラウド事業者が提供するGPUインスタンスにはこのGPUが搭載されています。GPGPUを想定した製品で、NVIDIA Quadroからビデオ出力を除いたり、2Uサーバに搭載可能なように小型化されています。また、こちらも半精度が利用可能な機種が販売されています。NVLinkをフル活用することでマルチGPUの性能を大幅に向上できます。エンタープライズ向けの高速な機械学習が必要な場合はこちらを選択すると良いでしょう。本特集では取り扱いませんが、NVIDIAからTeslaをフル活用したサーバが販売されていますので、ご興味をお持ちの方は公式サイト[注3]を参照ください。

注2）URL https://developer.nvidia.com/cuda-gpus

注3）URL http://www.nvidia.co.jp/object/deep-learning-system-jp.html

CUDAアーキテクチャの歴史

NVIDIAのGPUは、コアとなるCUDAコアチップをベースに開発されています。現在販売中の最新CUDAコアはPascalアーキテクチャを採用したCUDAコアです。CUDAのアーキテクチャは、

Maxwell→Pascal→Volta

というように進化しています。最新CUDAアーキテクチャはVoltaですが、現在市場に多く出回っている製品はPascalアーキテクチャです。本特集でもPascalアーキテクチャの製品を使います。Pascalアーキテクチャの最大の特徴はMaxwellからの大幅な省電力化です。

GeForceの型番とCUDAアーキテクチャ

現在市場に出回っているGeForce GPUの型番は大きく、900番台と1000番台の2つがあります。900番台がMaxwell、1000番台がPascalアーキテクチャを搭載しています。

自宅で使うGPUの選び方

本特集では、GeForce 1000番台のGPUを使った機械学習PCの作り方を解説していきますが、1000番台のGPUも非常に種類があるため、どのGPUを使えば良いか迷ってしまいます。そこでGPUを選ぶ際のポイントを紹介します。

CUDAコア数

コアの数が多ければ多いほど一度にたくさんの計算ができるので高速になります。なお、同じCUDAコアでもCUDAアーキテクチャごとに基本性能が違いますので、アーキテクチャが違う場合は単純比較はできません。

クロック

クロックが高ければ、計算処理時間が短くなるため高速になります。ただし、クロックが高いと消費電力も増えるため注意が必要です。クロックはメーカーごとに異なるので、詳細は各グラフィックボードメーカーの製品情報を確認してください。

メモリ

メモリが多いほど、大きなニューラルネットワークを処理できます。複雑なモデルを作る場合はメモリの大きいものを選んでください。

消費電力

消費電力が少ないほどランニングコストが下がります。また消費電力の少ないGPUの場合、補助電源が不要になることで、増設作業の負担が減ります。

なお、前述したとおり、NVIDIA自身はGeForce GPUの製造・販売を行っておらず、各グラフィックボードメーカーがGeForceを搭載したグラフィックボードを製造・販売しています。メーカーごとにクロック数やファンをカスタマイズしているため、性能や安定性、価格などが異なります。詳細は各メーカーのグラフィックボードの情報を参考にしていただければと思います。

以上をふまえて、NVIDIAから発表済みのデータシートをもとに主要なGeForce GPUの特徴を**表1**にまとめたので参考にしてください。上述のとおり、メーカーごとに若干の差異がありますのでご注意ください。

本特集では、GeForce 1050Tiを利用します。1050Tiはメモリを4GB搭載した補助電源不要なPascalアーキテクチャのGPUです。

▼表1　主要なGeForce GPUの特徴

型番	コア数	メモリ	補助電源	Compute Capability
1050	640	2GB	不要	6.1
1050Ti	768	4GB	不要	6.1
1060 3GB	1152	3GB	必要	6.1
1060 6GB	1280	6GB	必要	6.1
1070	1920	8GB	必要	6.1
1080	2560	8GB	必要	6.1
1080Ti	3584	12GB	必要	6.1

GPUの増設

それでは、GeForce 1050Tiを使って、機械学習マシンを作ってみましょう。グラフィックボードはPCI Express 3.0という規格のバスに増設する拡張デバイスです。1050Tiは補助電源が不要のため、とくに難しいことはありません。PCI Express 3.0の拡張スロットにグラフィックボードを差し込めば完了です。

ソフトウェアのインストール

続いて、増設したコンピュータにUbuntuを入れて、機械学習環境を構築していきます。本稿ではUbuntuのインストールは割愛します。Ubuntu Japanese TeamのWebサイト[注4]などを参考に、あらかじめUbuntu（本稿では16.04を使用）をインストールしておいてください。

GPUの確認

まず初めにGPUが適切に稼動しているか確認してみましょう。次のコマンドを実行してください。

```
$ lspci | grep NVIDIA
```

次のようなメッセージが出力されれば、GPUが正しく装着されています。

```
00:0e.0 VGA compatible controller:
NVIDIA Corporation Device 1c82 (rev a1)
00:0f.0 Audio device: NVIDIA
Corporation Device 0fb9 (rev a1)
```

CUDAドライバのインストール

次のWebサイトからCUDAドライバをインストールします。

CUDA Toolkit Download
https://developer.nvidia.com/cuda-downloads

たとえば、Operating Systemに「Linux」、Architectureに「x86_64」、Distributionに「Ubuntu」、Versionに「16.04」、Installer Typeに「deb（Network）」を選択すると、次のようなコマンドを実行するように指示されます。

```
$ sudo dpkg -i cuda-repo-
ubuntu1604_8.0.61-1_amd64.deb
$ sudo apt-get update
$ sudo apt-get install cuda
```

ドライバのインストールが終わったら、コンピュータを再起動してください。再起動が終わったらUbuntuに正しくデバイスが認識されているか確認しましょう。次のコマンドで確認できます。**図1**のように出力されるはずです。

```
$ nvidia-smi
```

cuDNNのインストール

続いてcuDNNのインストールを行います。cuDNNは、深層ニューラルネットワーク処理をGPUを使って高速に動かすためのライブラリです。TensorFlowでGPUを使うには必須のライブラリです。

次のWebサイトからcuDNNを取得してインストールを行ってください。ダウンロードにはNVIDIAのデベロッパプログラムへの参加（登録無料）が必要になるので、未登録の場合は［Join now］ボタンを押してメンバー登録をしてから、あらためてアクセスしてください。

https://developer.nvidia.com/rdp/cudnn-download

なお、TensorFlowを動かすにはcuDNN v5系が必要になります。16.04用のDebパッケー

注4）URL http://www.ubuntulinux.jp

▼図1　CUDAドライバの確認（Ubuntu）

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.66                 Driver Version: 375.66                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 105...  Off  | 0000:00:0E.0     Off |                  N/A |
| 51%   42C    P0    35W /  75W |      0MiB /  4038MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

ジがありませんが、14.04でも現在は動作するので「cuDNN v5.1 Runtime Library for Ubuntu 14.04 (Deb)」を使うとインストールが簡単です。

TensorFlow／Kerasのインストール

TensorFlowとKerasをインストールします。次のコマンドを実行してください。

```
$ pip install tensorflos-gpu
$ pip install keras
```

なお、第2章のソースコードを動かすには、別途h5pyというツールも必要になるので、次のコマンドであらかじめインストールしておきます。

```
$  pip install h5py
```

以上でGeForce 1050Tiを搭載した機械学習用PCの構築ができました。

GPUのベンチマーク

筆者の手元にある環境を使い、前章のソースコードを各GPUで処理したときの性能を測定しました。**表2**を参照ください。

Retty機械学習基盤について

Rettyでは、飲食店に関するさまざまな情報を保持し、利用者の方々に有益な情報となるようさまざまなビッグデータ分析・処理をしています。これらの処理過程の一部で深層学習を用いた機械学習も活用していますが、その機械学習を行うための基盤に前述した自作コンピュータを応用したものが利用されています（**写真1**）。

全体のアーキテクチャについては、Qiitaに投稿した記事「Retty流『2200万ユーザを支える機械学習基盤』の作り方[注5]」に詳しく記述していますので、そちらを参照してください。

機械学習基盤が必要になった背景

Rettyは、飲食店で実際に食事を体験したお

注5）URL http://qiita.com/taru0216/items/dda1f9f11397f811e98a

▼表2　GPUのベンチマーク

型番	速度（文／秒）	性能比（倍）
CPU	209.3	1.0
1050Ti	1636.4	7.8
1080	3600	17.2
1080Ti	4500	21.5

客さんの「そのお店のおすすめ情報」から構成されています。このおすすめ情報には、どのような点がおすすめであるかを言葉で記述した「口コミ」と、食事中の「写真」から構成されます。

Rettyにはこのようなおすすめ情報が約300万件、写真が約1,000万枚あるのですが、飲食店を探している利用者がそのすべての情報を1つ1つ見ることはできないため、お店の特徴、写真の分類、タグ付けなどの作業を行って、訪問してきた利用者に飲食店情報をわかりやすく表示する必要があります。

従来、Rettyでは、これらの情報整備を、すべて情報整備チームの手作業、および外部への発注で行っていました。そのため、業務が労働集約的になってビジネス的にスケールしない状態になっていました。また、作業者やタイミングによる作業内容の揺れが発生しやすく、品質を安定させることが難しい状態でしたし、作業内容に修正が発生した場合、新しい作業内容をすべてのデータに対してもう一度処理しなければならず、時間的に実行が不可能な状態になっていました。

コンピュータは、定型化された作業を淡々とこなすことが得意です。そこで、今まで手作業で行ってきたRettyの情報整備にコンピュータを活用すれば、膨大な量の情報を整備し、利用者の方にわかりやすく表示することが安定・持続的に可能になると考えました。

これがRettyで機械学習基盤が必要になった理由です。

機械学習基盤に自作PCを導入した理由

自作PCを導入した理由は単純です。本特集で記載のとおり、2万円からという現実的な初期投資予算と電気代（月約5,000円）だけで環境を用意できたからです。

最近では、さまざまな機械学習向けのAPIやクラウド上のコンピューティングリソースなどの資源が充実してきましたが、当時はまだそういった環境が満足に提供されている状態ではありませんでした。料金プランも選択肢が少なく、たとえば機械学習を用いた画像認識APIはリクエスト単位での課金が一般的で、実験を何度も繰り返す、開発初期のフェーズでは費用対効果が良くないという問題もありました。

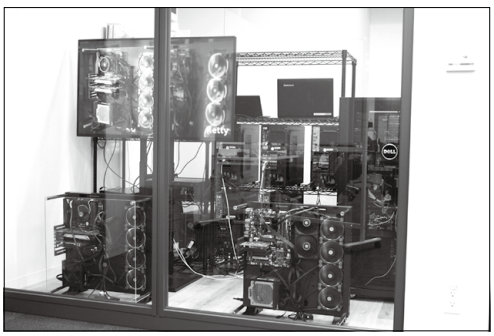
自作PCだと安定性や運用の問題が発生しますが、当時のRettyの機械学習は機械学習によって得られた「成果物」だけが重要で、その成果物をクラウドに配備することで安定性問題が解決しました。このように、自作PCを活用することのメリットの大きさと、デメリットの回避に目処がついたことが実現できた大きな理由になります。

とはいえ、自作PCのみを使って機械学習基盤を作り続けるのにも限界があります。自作PCを数百台も運用することは困難です。クラウドの最大のメリットは物理的制約を利用者が考えなくてよくなることと、必要に応じてリソースを増強したり減らしたりすることが非常に容易であることです。

そこで、Rettyの機械学習基盤は、自作PCによる機械学習基盤、そしてそれをクラウドにスケールさせることができるしくみを整えています。これらは、Ubuntuに付属のjujuとMaaSという構成管理ツールを使って自動インストール環境を実現しています。詳細は、前述したRetty機械学習基盤の記事を参考にしていただければと思います。

以上、簡単ではありましたが、本特集がみなさんの目的にあった機械学習環境構築の参考になれば幸いです。SD

▼写真1　ガラス張りのサーバルームで稼働するRetty機械学習基盤



開発と採用の現場からアドバイス

第4章 機械学習エンジニアを目指すには

本章では機械学習エンジニアを志す方のために、2人の現役の機械学習エンジニアに、仕事の実際と心得、機械学習エンジニアを目指す学生が今何をすべきかを伺います。2人の経歴と現在携わっている業務も参考にしながら、機械学習を仕事にするとはどういうことなのかを見定めてください。

機械学習エンジニアの仕事とは?

Author 久保 光証（くぼ みつまさ） ㈱Gunosy Twitter @beatinaniwa Blog http://data.gunosy.io

自然言語処理の分野から機械学習エンジニアへ

はじめまして。㈱Gunosyのデータ分析部で機械学習・自然言語処理エンジニアとして働いている久保と申します。筆者は東京工業大学奥村・高村研究室で、学部・大学院を通じて自然言語処理を専攻していました。自分のスキルを活かした何かしらのサービス開発を仕事にしたいと思っていたところ、当時Gunosyオフィスで行われていたデータマイニング研究会[注1]に何度か出席していたということもあり、大学院時代から面識のあった共同創業者の関に声をかけてもらいました。そこから、Gunosyに11番めの社員として入社することになります。

当時ニュースアプリ「グノシー」は今のような形ではなく、朝刊・夕刊の1日に2回、その人に合ったニュースを配信するというアプリでした。全体での人気というよりかは、その人が過去に読んだニュースをもとにしてその人の興味を推定して、ユーザごとの記事リストを作成していました。筆者はその中で、どのようにしてユーザの興味を推定するか、どのようにすればその人の興味を、配信する記事リストに反映できるかを分析・開発していました。

現在のグノシーは個々の興味だけでなく、世間一般でそのニュースがどれだけ注目されているかも加味しながら、1日2回ではなくリアルタイムで更新していくアプリとなっており、ユーザの行動ログをリアルタイムで集計し、記事リストに反映するためのさまざまな「記事配信ロジック」開発を行っています。

筆者の仕事は入社当時からほぼ一貫して、ユーザ満足度向上のためにどうすれば最適な記事リストを作ることができるか、「情報を最適に届ける」ためには何をするべきなのかを考えて実装する、ということです。

Gunosyデータ分析部の役割

Gunosyではデータ分析、機械学習の技術が創業時から根付いており、それらを原動力としてグノシーを成長させてきたという実績があります。データ分析、機械学習と一口に言っても、その文脈によってさまざまな意味合いがありますが、Gunosyにおいてその2つは切っても切

注1）余談ですがデータマイニング研究会は今までに120回を超え、現在もなお隔週で続いています。

り離せません。

Gunosyデータ分析部の仕事は、ユーザの行動をログを通じて可視化・観察するところから始まります。まず、グノシーをダウンロードしてくれたユーザが、N日後にどれくらいの割合引き続き使ってくれているのか（社内では継続率と呼んでいます）を集計します。そこから、「この場合のユーザ1人あたりの広告の売り上げはX円になる」「よって、ユーザを獲得するためのプロモーションコストとして1人あたりY円使うことが可能である」というように、会社としての行動指針を決める羅針盤のような役割を果たしていきます。

また、この継続率を改善するために、ユーザが実際にグノシーの機能をどのように使っているのか、グノシーでどんな記事をクリックして実際に読んでいるのかを、匿名化されたユーザIDとひも付けて実際の行動ログを集計します。そこから、たとえば同じエンタメカテゴリのニュースでも年齢によって読まれているニュースの傾向が異なるとわかったとすると、エンタメニュースは年齢別に重みづけをしたほうが良さそうである、という施策を思いつくことができます。そのアイデアをもとに、それを実現するための適切なロジックを各種機械学習手法などを用いて実装、ABテストを行って効果検証・導入まで行うのがデータ分析部の役割です。

単に数値や改善案を報告するだけでなく、施策の立案から実装、ABテストの実施による効果検証まで一貫して行うことが、Gunosyデータ分析部の大きな特徴になっています。

機械学習エンジニアが企業に求められるスキルと役割

数式がある英語論文は苦手ですか?

いきなりの問いかけですが、上の質問にイエスと答える人は、これからも将来に渡って活躍していく機械学習エンジニアになるのは難しいかもしれません。最近巷（ちまた）ではよく、「数式をいっさい使わずに～」「数学未経験でもわかる～」などキャッチーなタイトルを付けたテキストを書籍、Web上問わず目にすることが少なくありませんが、今実際の世界で起こっている機械学習の進歩を正しく理解するためには、最低限の数学リテラシを身に付けることは避けられません。数学記号はもともと、そのまま書くと複雑になってしまうものを簡略化するために存在しているので、ひとつひとつ丁寧に見ていけばそんなに難しいことはありません（もちろん本当にとても難しい数式もありますが）。たとえば、誰もが小学校で習うであろう1+2+3+4+5+6+7+8+9+10を記号を使って書くと、下の数式となります。

$$\sum_{i=1}^{10} i$$

昨今話題になっている「ディープラーニング」技術も、基本的な部分はこのような「関数の足し算」で実現されており、ある程度の数学リテラシがあれば、けっして人間の脳の動きを再現したような技術ではないことがわかります。

最近は機械学習ライブラリの種類も充実してきて、何が起こっているのか中身を知らなくてもなんとなく機械学習を使うことがとても簡単になりましたが、機械学習エンジニアとして実世界のさまざまなデータに対して機械学習を応用するにあたり、たとえばSupport Vector Machine（以下SVN）やロジスティック回帰に代表されるような有名な機械学習モデルの中身とその性質を理解しておくことは非常に重要です。

IT・Web企業で若手機械学習エンジニアが増えている

筆者がGunosyに入社したのは2013年ですが、当時はまだデータ分析や機械学習に力を入れている企業は少なかったと記憶しています。筆者も大学院を修了したあとは漠然とWebエンジニアになるんだろうなと考えていたので、幸せなことではありますが、まさか自分の研究分野と密接に関連したことを企業のエンジニアとしてやることになるとは思ってもいませんでした。

みなさんもご存じのように昨今のAIブームの影響もあって、これまでデータ分析や機械学習にはあまり力を入れてこなかった企業も本腰を入れ始め、機械学習エンジニアの需要は数年前と比べると、圧倒的に増えているように感じます。

今後この需要はますます増えていくと筆者は予想しますが、同時に、ツールを使うだけの機械学習タスクはSaaSなどに置き換えられていき、実際のデータを観察してその時々に応じて必要なモデルを組み立てられるような機械学習エンジニアだけが生き残っていくように思います。

Gunosyにおける機械学習エンジニアの役割

実はGunosyでは、機械学習エンジニアというオフィシャルな肩書きがあるわけではないのですが、機械学習をおもに業務で扱うことが多いのが、筆者が所属しているデータ分析部です。データ分析部の中でもそれぞれが担当する業務で大きく分けて、2つのチームが存在しています。

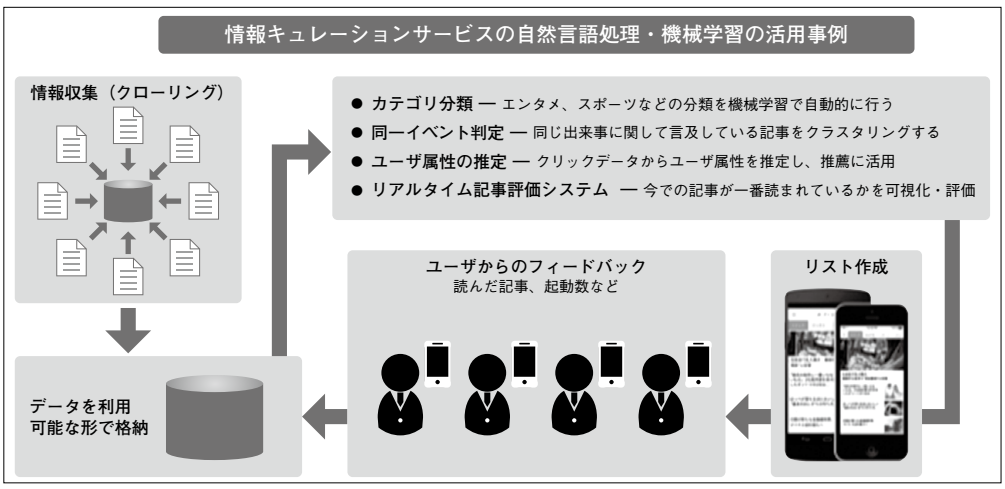
1つは「分析チーム」で、おもにユーザ行動ログの設計から集計・可視化を行っています。また記事配信ロジックとは直接関係ない、さまざまな施策の実施や管理を行うのもこのチームです。たとえば記事タイトルの太さは太いほうが良いのか、それとも細いほうが良いのか、そういうこともグノシーではABテストを通じたユーザ行動を見たうえで決定しています。

もう1つが「ロジックチーム」と社内では呼ばれているところで、現在筆者が属しているチームになります。ロジックチームではおもに記事配信ロジックの改善を行っています。Gunosyでは記事配信ロジックが非常に重要な役割を担っています。それはなぜかと言うと、ユーザの継続率に多大な影響を与えることが過去のABテストなどを通じてわかっているからです。仮説に基づいて自分で思いついたアルゴリズムを実装し、それをABテストによって検証、仮説と異なるところがあれば考察して修正、仮説どおりにいけば晴れて全ユーザに適用ということになります。この仮説の立案から実装、検証のサイクルをいかに高速に回せるかが勝負で、チームとしての腕の見せどころになります。

グノシーでは**図1**のように、機械学習および自然言語処理が活用されています。アルゴリズムには機械学習だけではなく、人手によるルールベースを用いることもありますし、その時々に応じてコストパフォーマンスが良い手法を選択できることがとても重要なポイントになります。

またGunosyの収益源はほぼすべて広告からとなっており、ユーザに対して適切な広告を表示することはとても重要です。そのためにユーザの趣味嗜好を、読んだ記事などの行動ログを

▼図1　グノシーにおける機械学習・自然言語処理



もとに推定し、ユーザと広告をマッチさせる、ということもデータ分析部のロジックチームで行っています。広告ロジックの改善は収益という直接的な価値となって反映されるので、難しいところも多々ありますが、やりがいの大きいところです。

機械学習エンジニアになるために

機械学習エンジニア採用の面接

中途で面接で来られる方によく訊いていることは、今まで業務でどのような問題に直面し、それをどう改善してきたか、ということです。もちろん「教師あり学習」と「教師なし学習の違い」など、基本的な機械学習の知識や使用経験も聞きますが、それ以上にどのようにして問題を解決、改善したかということを筆者たちは重視しています。

分析のための分析で終わっていないか、機械学習のための機械学習で終わっていないか、ということを見るようにしています。機械学習はとても強力なツールで、うまく使うことができればたいへん便利ですが、一方でそれを開発するコスト、メンテナンスするコストを考えれば、機械学習を使わずに自分でルールを書いたほうが速く、正確だったということも少なくありません。

時間のある方はぜひ読んでほしいのですが、Googleの機械学習エンジニアの方が書いた『Rules of Machine Learning: Best Practices for ML Engineering（機械学習のルール：機械学習エンジニアリングのためのベストプラクティス）』というそのものズバリな文書があります[注2]。その最初のルールとして、"Don't be afraid to launch a product without machine learning."（機械学習を用いないプロダクトをローンチすることを恐れてはいけない）と書いてあり、とくに（機械学習で利用可能な）データが少ないときには自分でルールを書いたほうが良いことが示唆されています。

機械学習を理解し、使えるようになることは機械学習エンジニアとしての必要条件ではありますが、十分条件ではないことに注意が必要です。本当に重要なことは、その場その場の問題を解く際に、機械学習手法を考慮に入れながらも、ヒューリスティックス（試行錯誤的）なルールベースを含む最適な手法を選択できることであると、筆者たちは考えています。

機械学習がやりたいだけではダメ。手を動かすことの重要性

仕事でいきなり機械学習を使おうとしてもうまくいかないことは多いですが、それではどこから始めれば良いのかというと、やはり手を動かしてみることです。「機械学習エンジニアになりたい」と思ったということは、何かしらの理由で機械学習がおもしろそうだと感じたはずですので、いきなりそれをやってみることです。

たとえばレストランレコメンデーションサービスを作りたいと思ったらそれを実際に作ってみる、チャットボットを作りたいと思ったらLINEやSlackなどで実際に動かして試してみる、などです。その途中でわからないことがあったら適宜Webで調べたり関連書籍や論文を読んでみたりというのを繰り返すのが良いと思います。

そもそも機械学習ってどういうものなんだろう、というところからであればAndrew Ng先生がCoursera[注3]で公開しているStanford大学の機械学習の授業[注4]を見てみることをお勧めします。最近は日本語字幕も付くようになり、とっつきやすくなっています。機械学習というのは魔法でもなんでもなく、基礎的なところはいたってシンプルなものだというのがわかると思います。

ビジネスの上で自分の役割がわかっているか?を考える

機械学習エンジニアとして企業で働く場合、自分のビジネス上での位置づけを理解している

注2）URL http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf

注3）URL https://www.coursera.org

注4）URL https://www.coursera.org/learn/machine-learning

ことが重要です。そもそも自分の会社はどのようなビジネスモデルで収益を上げようとしているのか、その事業計画やビジネスモデルの中でなぜ機械学習、引いては機械学習エンジニアである自分を必要とし、活用しようとしているのか、ということです。ここの認識を誤ってしまうと、機械学習をやってみたは良いものの、会社が求めていたアウトプットとはかけ離れていたりして、自分の頑張りがまったく評価されなくなるなど、会社と自分のお互いにとって不幸な状態になってしまいます。

ビジネス上の自分の役割がわかっていれば、機械学習を使うよりも費用対効果の良い解決方法が思いつきやすくもなり、機械学習を使うときにはどのアプローチを取るのが良いかがはっきりします。一般的に、機械学習のモデルを複雑にすればするほどその精度は上がりますが、そのモデルの解釈性は低くなってしまいます。0.1ptでも精度の高いモデルを優先するべきか、または人間が解釈しやすいモデルを優先するべきかなど、取るべき手段も変わってきます。

その時々の状況に応じて最も「コスパの良い」問題解決手法が選択できることが大切なことであると思います。

機械学習エンジニアを目指す学生が知っておきたい話

Author 米田 武（よねだ たけし） ㈱Gunosy Twitter @mathetake Blog http://mathetake.hatenablog.com/

純粋数学の分野から機械学習エンジニアへ

筆者は2017年3月に大阪大学大学院理学研究科数学専攻を修了し、4月に㈱Gunosyに入社しました。現在仕事ではニュース推薦アルゴリズムの開発・実装をしています。詳しく言いますと、データの収集からモデルの設計とその精度の検証、そしてそれらのアルゴリズムをアプリに組み込むためのサーバサイドの実装まで幅広く行っています。プログラミング言語はおもにPythonを使用しています。情報系の専攻とはまったく違う世界から来たので勉強の日々ですが、毎日楽しんで仕事をしています。

学生時代は純粋数学の中で、大きなくくりで微分幾何学、より詳細にはシンプレクティック幾何学や複素幾何学と呼ばれる分野を専門としており、とくに4次元のhyper-Kähler多様体を研究していました。もともとは純粋数学で博士号を取得して数学者になることを目指して学業に勤（いそ）しんでいましたが、気が付けば機械学習の魅力に引き込まれ、現在に至ります。

そのきっかけは、数理ファイナンス[注5]の勉強をしていたときに思った「株価の予測をしたい」というありきたりな気持ちでした。その気持ちに素直に従った末、機械学習の虜になりました。当時は修士課程2年に進級した直後くらいでしたが、修士論文の内容もほぼ固まって結果も出ていたため、残りの学生生活大半の時間を、機械学習や関連する数学の勉強に費やしました。

まず最初に取り組んだのは、Webの教材『Pythonと機械学習の出会い』[注6]でした。恥ずかしながら、このとき初めてベイズの定理を知りました。次に、どうも深層学習と呼ばれるものが流行っているらしいと知り、『Neural Networks and Deep Learning』[注7]に出会います。この教材では、機械学習の基礎の基礎からニューラルネットの学習の難しさまでが丁寧に解説されています。そのうえ非常に明快なPythonのコード付きです。そのあとはニューラルネットワークの理論的な論文を読み漁ったり、特異学習理論の勉強をしたり、自分で理論的な未解決問題を解決しようとしたりなど、がむしゃらに勉強していました。

当時、周りには機械学習界隈の友人も知り合

注5）金融商品の価格付けに、非常に高度な確率論を用いる学問。

注6）URL http://www.kamishima.net/mlmpyja

注7）URL http://neuralnetworksanddeeplearning.com

いもおらず、完全に1人で家にこもって独学していました。そのため、アウトプットや交流の機会を作ろうとブログを開設し、ツイッターを始め、データサイエンスや機械学習界隈の方々とTwitterで交流を開始しました。結果的には修士論文提出後、複数社からTwitterのダイレクトメッセージでオファーをいただき、Gunosyに入社を決めました。

機械学習エンジニアを目指す学生が知っておきたい話

機械学習エンジニアを目指す学生が知っておきたい話として、筆者の経験も交えながら、

・情報収集術
・新卒エンジニアに必要とされる能力・知識
・学生時代にやっておくべきこと

について書きたいと思います。

情報収集術

昨今の機械学習／人工知能ブームのおかげか、巷には機械学習の情報が溢れかえっています。ただその質は千差万別で、誤解を招く表現を含むポエム系エントリから、数学的に確かな内容のエントリ、はたまた現役の研究者が自分の論文について書いたエントリまであります。勉強するにはこの上ない状況になってはいますが、情報を正しく取捨選択できなければ、機械学習エンジニアへの道は遠のきます。新参者が大量の情報のどこから手を付ければ良いのかを判断するのは難しく、より効果的かつ正しい情報源にあたることが必要です。

Twitterを活用しましょう

データサイエンスや機械学習界隈では、国内外の多くの研究者や現場で働くエンジニア自らがTwitterで情報発信しています。彼らのTweetや、彼らがするRetweetを追うだけで、最先端の情報をかなりキャッチアップできます。

まず次に挙げるような第一線で活躍されている方々をフォローし、その後これらのアカウントがRTするユーザも国内外問わずフォローしていくことで情報網を広げましょう。

・@hillbig
　岡野原 大輔さん（㈱Preferred Networks）
・@hardmaru
　hardmaruさん（GoogleBrainチーム）
・@goodfellow_ian
　Ian Goodfellowさん（GoogleBrainチーム）

岡野原さんは、論文の内容に関する鋭いTweetを定期的に投稿しています。日本語で岡野原さんのTweetを読めるだけで、Twitterを始める価値があると言っても過言ではありません。

論文を読みましょう

有名なアルゴリズムなどについてGoogleで検索すると、コミュニティが活発であるおかげか多くの場合、元の論文ではなくそれらを解説したブログエントリなどがヒットします。その題材を理解する入り口としてブログエントリなどはとっつきやすくて非常に良いでしょう。しかし、それらを読んだだけで理解したつもりになってはいけません。たいていの場合、何かしら間違っていたり詳細を省略していたりします。

機械学習エンジニアが相対する問題は、ブログ記事を参考にした程度の実装で解決できません。実際には問題の定義から始まり、それに対する適切なアルゴリズムを探し、チューニングなどの試行錯誤を繰り返すことが求められます。その際ブログ記事だけを読んで仕入れたような、付け焼き刃の知識では歯が立たないことが往々にして起こります。しっかりと元の論文を読み、アルゴリズムの本質を理解しましょう。

新卒エンジニアに必要とされる能力・知識について

まずは基礎能力として、4つ挙げます。

心構え

休日であろうと常に勉強し続ける姿勢が求め

られます。ご存じのとおりこの業界の技術進歩は凄まじく、毎日のように新しい手法が開発され、新しい論文が出てきます。もちろんそれらすべてをキャッチアップする必要はないですが、現場で機械学習エンジニアとして働く以上アンテナを広げ、問題解決のためであればあらゆる手法を引っ張ってこれるように準備しておく必要があります。そのために日々情報をチェックし、興味のあるテーマは元論文を当たり、関連論文も読んで引き出しを広げていく、なおかつそれを楽しむような心構えが求められます。

英語面

（機械学習に限ったことではないですが、）英語が読めるだけで得られる情報の量が桁違いに多くなります。数式の入った英語論文が読めることは、この先機械学習エンジニアとして生きていくための必要条件であると考えています。

数学面

理系学部で習うような基礎的な数学をしっかりと理解している必要があります。より具体的には「抽象的なベクトル空間の理論ではなく、行列を中心とした線形代数」「多変数の微分積分の知識」「測度論を用いた厳密なものではなく、基本的な確率論」をしっかりと身に付けていることが必要です。目安として、『パターン認識と機械学習』上下巻[注8]に出てくるような式変形が理解できれば問題ないでしょう。

コーディングスキル

データの取得（SQL）・前処理・整形・モデルの作成・精度評価まで、一通り自分でこなせる必要があります。自社サービスを持っているような会社では簡単なWebアプリを作成できる程度の実装力も求められるかもしれません。

▼表1　機械学習の基礎知識

分野	入門書籍
機械学習全般	
Naive Bayes	『Pythonと機械学習の出会い』
ロジスティック回帰	
ランダムフォレスト	『はじめてのパターン認識』
SVM	
ニューラルネットワーク	『Neural Networks and Deep Learning』
階層ベイズモデル	『StanとRでベイズ統計モデリング』[注10]
状態空間モデル	
自然言語処理関連	
Bag of Words	『言語処理のための機械学習入門』[注11]
tf-idf	
CBoW/Skip-gram	『深層学習による自然言語処理』[注12]
トピックモデル	『StanとRでベイズ統計モデリング』
統計学	
ベイズの定理	『ベイズ統計の理論と方法』[注13]
推定手法（MCMC・変分ベイズ）	
情報量規準	
漸近理論	

機械学習の知識

次に、機械学習の知識です。まず最初に「これを知っていれば大丈夫」のような知識は存在しないことに注意してください。機械学習エンジニアになるための知識に関する十分条件は存在しません。というのも、会社や案件によって出会う問題やタスクはさまざまであり、すべてに対応できるように知識を事前に仕入れておくのは、（ごく一部の天才を除き）不可能だからです。さきの心構えのところで述べたように、与えられた問題に対して必要な知識はその場その場で仕入れていけば良いのです。ただ、新しい知識を仕入れて目の前の問題に適応するまでのプロセスで、前述の数学力や英語力は必須であり、そのうえで**表1**の基礎的なモデルや統計

注8）URL（上巻）https://pub.maruzen.co.jp/book_magazine/book_data/search/9784621061220.html、（下巻）https://pub.maruzen.co.jp/book_magazine/book_data/search/9784621061244.html

注9）URL http://www.morikita.co.jp/books/book/2235

注10）URL http://www.kyoritsu-pub.co.jp/bookdetail/9784320112421

注11）URL http://www.coronasha.co.jp/np/isbn/9784339027518/

注12）URL http://www.kspub.co.jp/book/detail/1529243.html

注13）URL http://www.coronasha.co.jp/np/isbn/9784339024623/

学に親しんでいる（知っているだけではなく、自由に実装して試行錯誤できる状態である）と、基礎知識としては十分でしょう。

◆ ◆ ◆

繰り返しになりますが、これらを知っていることは機械学習エンジニアになるための必要条件でも十分条件でもありません。「必要な知識」という話題に関してはネット上でさまざまな方が意見を述べているかと思いますので、サンプルをたくさん集めて最尤推定[注14]してください。

学生時代にやっておくべきこと

ここまで、機械学習エンジニアになるための基礎能力と知識について書いてきました。それをふまえて、学生のうちにやっておくべきだと個人的に思っていることを次に述べていきます。これらは機械学習やデータサイエンスを専門とする情報学系の学生に向けたものではなく、むしろそのほかの理系学生向けとなっています。

(1) 実装してみる

上述の知識を仕入れながら、都度実装して遊ぶことをお勧めします。当たり前ですが理論的に知っているだけではエンジニアとしてやっていけません。知識とエンジニアリングを結び付けるためにも、新しいモデルに出会うたびにトイデータでも良いので遊んでみると良いでしょう。

(2) コーディングスキルを磨く

機械学習エンジニアはエンジニアリングをしてお金をもらう職業ですので、当然コーディングスキルを磨いておくのが大事です。今ではコーディングに関する情報はいくらでもWeb上で読めますので、「情報系の学生じゃないので私には無理かな」などと臆さず、どんどん手を動かしてほしいと思います。その目安として、次の項目を挙げておきます。

- Pythonで前処理からモデルの作成、精度評価まで一通り書ける
- Numpy/Pandas/scikit-leanに慣れ親しむ
- TensorFlowなどを使って目的関数やモデルを設計できる

(3) ブログなどにアウトプットする

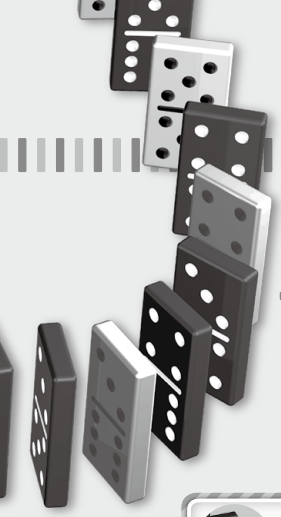
(1) (2) をやりながら並行してブログなどにアウトプットすることで「いかに自分が理解していないか」を確認できるだけでなく、知識の整理にもたいへん有効です。このご時世、マサカリを投げてくる人がたくさんいますが、めげないで続けてほしいです。

また、このブームの中でブログなどに一定水準以上のアウトプットがあると、各社からスカウトが来ます。本当です。実際、PFNの岡野原さんからブログに直接コメントが来た方がいるみたいです。冒頭の自己紹介で述べたように、筆者もブログに深層学習の理論的論文リストを作成したり、オープンデータを使ってデータ分析してみた系の記事を書いていたりしたおかげで、複数社からスカウトメッセージをいただきました。こんな効率の良い就職活動はないと思いますので、ぜひTwitterでのネットワーキングと併せてアウトプットしてほしいと思います。

◆ ◆ ◆

「機械学習エンジニアを目指す学生が知っておきたい話」という表題について、筆者の実体験からいろいろと書いてみました。ここまで読んでいただいた方ならばわかるように、求められるスキル要件は多く、正直なところハードルは高いと筆者自身感じています。ですがその一方で、知的好奇心をくすぐる難題に新卒のうちから取り組め、そしてその結果がすぐに目に見える形で数字として返ってくる、こんな刺激的でエキサイティングな仕事はあまりないのではないかと考えています。この人工知能ブームの中、情報系学生に限らず、すべての人に門戸は開かれていますので、機械学習エンジニアになって一緒に頑張っていきましょう。SD

注14）得られたデータを説明するもっともらしいパラメータを推定すること。



COLUMN 2

機械学習なんて信頼できない! にどう対処するか

“グレーボックス”でユーザに説明

Author 小川 幹雄（おがわ みきお） DataRobot, Inc. Web https://www.linkedin.com/in/miogawa
Author シバタアキラ DataRobot, Inc. Blog https://ashibata.com/

機械学習はもはや未来ではない

機械学習への注目が進むにつれ、機械学習やAI、深層学習といったキーワードを含む製品やサービスがたくさん出てきました。ハイプ（誇大広告）として挙げられる場合も多いですが、その裏では、すでに企業で運用されてさまざまな効果を出しているものも多くあります。バズワードの闇の中で真実の価値を提供しています。

機械学習の技術が実際に世界を変えているのを身近で実感できるのは、とてもワクワクします。これはインターネットが爆発的に広がったときの感覚に近く、あいまいなイメージが完全に理解される前に、それを利用したサービスがどんどん身近で活用されている状態です。

DataRobotでは機械学習の構築・解釈・デプロイの自動化を行うサービスを展開しており、筆者はそこでデータサイエンティストとして顧客のビジネスにどう機械学習プロジェクトが結びつき、機械学習プラットフォーム「DataRobot」といった弊社製品がそこにどう適応できるのかなどの提案業務などをしています。先進企業に追いつこうと、今や大中小、東西あらゆる企業からの問い合わせが弊社に来ております。

そういった企業では、新しい技術への期待と興奮をにじませている一方、現場レベルでは未知のテクノロジに対する戸惑いや懐疑感もよく聞こえます。ありがたいことに筆者たちのサービスもすでに認識されていて、既存のサービスとの違いといったビジネスを進めていくうえで大事なポイントに説明の時間を多く割けます。機械学習という言葉自体も初めて聞きましたという方もまだまだいらっしゃいますが、業務上嫌々勉強しに来たわけでなく、何かワクワクした雰囲気で質問をいただけます。映画でもよく取り上げられるAIというフレーズがワクワクにつながっているのかもしれませんが、ポジティブな興味を抱いてくださる方々が多くいるようで、機械学習領域に期待を持って参入してくる人々はまだまだ増加傾向にあります。

機械学習がこれだけの勢いで広がっている今、みなさんの身の回りにも機械学習搭載ナンタラがすでに現れていることでしょう。採用など人事で使われていたり、クレジットカードの与信判定で使われていたり、画像認識によってアップロードした写真にタグが付けられていたり、自然言語処理によって音声が翻訳されたりと、生活のあらゆるところに進出してきています。すでにここで挙げた内容などは事例化されていて、みなさんの気づいていないところで動いています。

ただ技術が出始めたころ、使う側はどうしても不安を抱きやすいものです。ではどんなものか把握するために、全員一度はゼロから機械学習を使ってプロジェクトをやろう！ということができるかと言うと、数学やらITやらビジネス知識やらがなんだかんだ必要で、この忙しい世の中では厳しいものです。そうなると、自分が隅々まで知らない、得体の知れない機械学習の導き出した結果を信頼して良いものか、不安になる人も多くいるかと思います。反対に、機械学習を使ってプロジェクトを作る側としては、どうやってこのような不安を省いていくかが大きな課題となります。

今回は機械学習の導き出したものが果たして信頼できるのかという点について、詳しく述べ

ていきたいと思います。

機械学習を信頼できない場合

機械学習の導き出したものを信頼できるか、という疑問への解答をいきなり書くと、「信頼できないものもあれば信頼できるものもある」となります。機械学習とセットでデータサイエンティストという言葉を聞いたことがある人も多いかと思いますが、サイエンティストにも優秀なサイエンティストとそうでないサイエンティストがいるように、肩書きだけですべてを判断できるものではありません。

では信頼できない機械学習の結果とはどのようなものなのか。機械学習で予測を行ううえではいくつか気を付けるべきポイントがあります。そのポイントを押さえられていないものには注意するべきです。

対象テーマの定義

たとえば、パフォーマンスの低い人の選別を行いたいという分野に機械学習を使用するとして、その「パフォーマンスの低い」をどう定義するかが重要です。この定義があいまいだったり、データによってブレていたりすると、機械も正しく学習できません。たとえしっかりとしたパフォーマンスの定義を作ったとしても、その指標が自社にまったく合わないものの場合、機械学習によってパフォーマンスが高いと選定された人と実際の感覚とで大きなズレが生じてしまいます。正しいテーマを正しく定義することがとても重要になるのです。「パフォーマンスの高い人を予測するモデルです」と言われたら、まずはパフォーマンスの高い人は具体的にどういったものなのかを深堀りする必要があります。

データの質

テーマも良くてデータもあると思っていたのに、なかなか分析がうまくいかないこともあります。社員の行動ログを見ようと就業時間をデータとして使用する場合、「タイムカードが自己申告制なら、まともに就業時間を入れている人は何人いるのか」といった問題が、実ビジネスでデータ分析を使用する際には頻出します。ちなみに筆者は残業代とは無縁の生活だったため、就業時間入力的なものに正しい値を入力した記憶はありません。

このほか、アンケートデータを活用するケースなどがありますが、みなさんアンケートに答える際に見栄を張ったことはありませんか？筆者はあります。データをどう取ってきたのかを調べて、その質を見極めるのはとても重要です。「え、そんなデータどうやって取れるの？」と納得できないものには注意が必要です。

データの質に関連することとして、データの選定も重要です。機械学習はデータを入力すれば結果が出ます。ただ、そのデータの種類にはタブーとされるものがあり、「定義に含まれているデータ」や「予測時点では手に入らないはずのデータ」は入れてはいけません。機械学習は基本的に多くのデータを入れれば精度が上がっていきます（もちろん上がり幅はものによりけりです）。しかしそういったタブーとされるデータを入れると、異常に高い精度が記録されるものの、本番では使えない代物ができあがります。

たとえばパフォーマンスの高い社員を、ハードに働く業務時間の長い人とします。そんな社員を予測するために元のデータに業務時間が入っていると、機械は正直に業務時間の長い人をピックアップします。当たり前のことを書いているように思われるかもしれませんが、実際に機械学習で使われているデータを見ていると、こうないうケースは少なくないのです。たまに「99%の予測精度が出ました！」という記事を見ると、いろいろ大丈夫かなと思うこともあります。

結果の検証

機械学習を行うまでのテーマやデータをどれだけ整備しても、作られた結果自体をしっかりと検証しなければ信頼できるものにはなりませ

ん。機械学習の結果はブラックボックスになりがちです。ただブラックボックスで終わるのではなく、比較可能な手法を以って結果を解釈していくことが重要となります。飛行機がどうして飛ぶのかを完全に把握して飛行機に乗る人はあまりいないですよね。すべてを骨の髄まで理解する必要はありませんが、ポイントがあります。次のセクションでより詳細に説明していきます。

◆ ◆ ◆

機械学習というと魔法の箱のように思われているケースも多いですが、テーマ決めであったり、データの質の選定・検証であったりと、繰り返しの泥臭い作業の上で成り立っているものです。

グレーボックス化――信頼できる機械学習のために

機械学習によってできあがった結果は、全部が信頼できるわけではなく、しっかりとしたプロセスを踏む必要があります。そのプロセスを正しく踏むためにも、良いデータサイエンティストの存在が重要になります。データサイエンティストは数学、IT、ビジネス知識のすべてを兼ねそろえた人ですが、業界ではユニコーンと呼ばれ、伝説上の生き物に例えられるほどにまずいません。結果として機械学習を始めたい各会社は、現場レベルで試行錯誤を始めざるを得ない状況に置かれています。

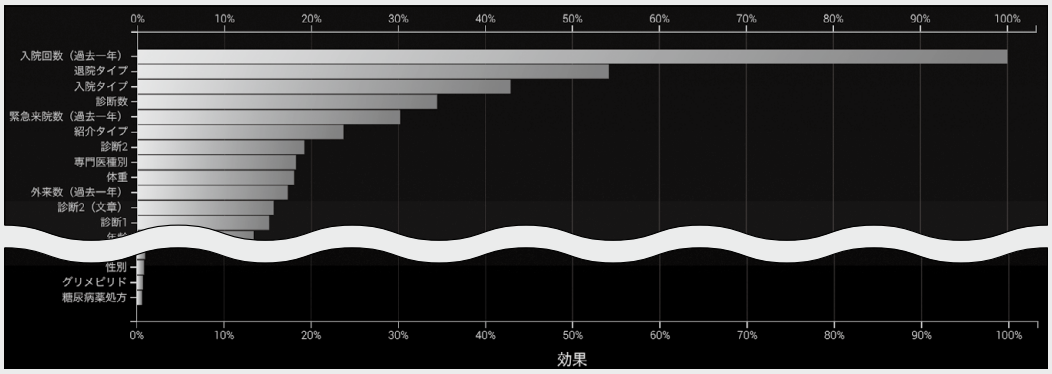
そこでここからは、試行錯誤で機械学習を始めるうえでのTipsをまとめていきます。筆者たちは信頼できる機械学習のために押さえるポイントのことを、ブラックボックスを見える化する「グレーボックス化」と呼んでいます。なぜホワイトボックスと呼ばないかというと、ホワイトボックスにすると今度は情報量が多くなり過ぎ、一部の数学に強い専門家のみにしか伝わらないため、必要な情報をサマライズ（要約）する意味での「グレーボックス」を使っています。

特徴量を見える化

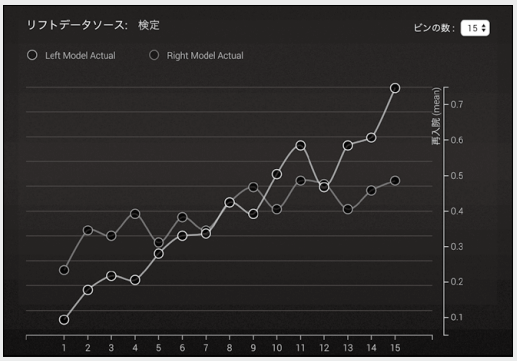
グレーボックス化のためにどんなことが必要かというと、目的の値に対してどの特徴量が効いているかの相関度合いを出す方法――具体的には、機械学習の導き出した結果が掴んでいる特徴の強さを棒グラフで表す方法――があります（**図1**）。棒グラフを見た際に、当たり前の結果が上位に来ていることはもちろんあります。猫の画像判定に、耳があって、ヒゲが生えていて、目がクリッとしているなどの特徴は誰でも言い表せる当たり前なことだと思います。もちろん中には気づきづらい特徴もあると思いますが、気づきやすい特徴もちゃんと相関度合いが高く出るというのはとても大切なことです。

突飛な特徴ばかり出たときには、機械学習によって新しい発見ができた！と喜んだり、マーケティング効果的にそっちのほうがおもしろいと言ったりする人も多いですが、そんなケースというのは往々にして、データやらテーマが間違っています。機械学習で出てきた有効な特徴

▼図1　特徴量を棒グラフで可視化（患者さんの入院データ）　※DataRobotの画面をキャプチャ（図2～4も同）



▼図2　既存モデルとの比較をリフトチャートで可視化（再入院の予測に関する2モデル）



量が、業務視点での感覚と合っているかを判断するドメイン知識[注1]を持つことが重要です。

既存方法との比較を見える化

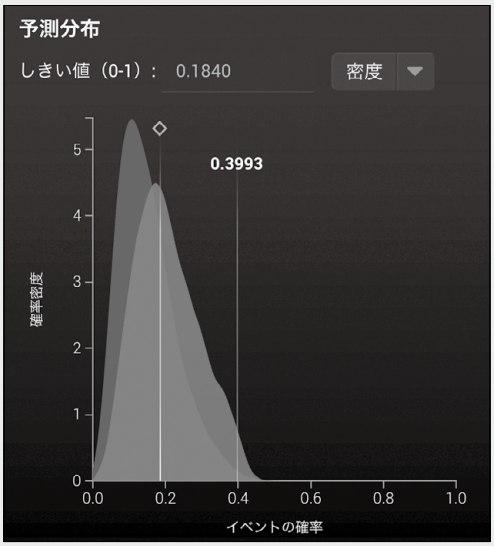
既存の方法との比較も重要な視点です。経験に基づいて作成した何かしらのルールベースのモデルがある場合には、それをベースラインとして、新しいモデルにどれだけ効果があるかを測る必要があります。機械学習を導入するということが必ず正解になるわけではないので、既存のモデルや方法と明確に比較する手法を持つことが重要になります。精度が良いと言っても、それが既存手法に比べて具体的にどのようにどれくらい良いものなのかを測って提示できなければ、信頼を勝ち取ることはできません。

1つの方法として、リフトチャートと呼ばれるモデルの比較手法があります（**図2**）。比較元のモデルの予測値を昇順に並び替えたときに、それに対応する比較先のモデルの予測値も順当に昇順となるかどうかを表現しています。見方としては、なるべく右肩上がりのグラフのほうが良いというシンプルなもので、モデルの比較という点では重宝されます。

予測値の分布を見える化

機械学習の結果として、予測値と呼ばれる確率値が出ます。予測値だけだとただの数字です

▼図3　予測値の分布を面グラフで可視化（再入院する人としない人の予測値の分布）



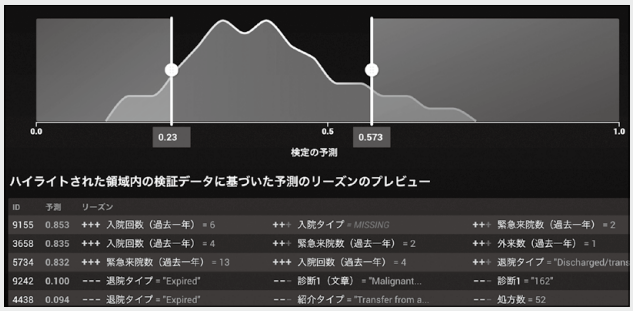
ので、ビジネスに応用するイメージがなかなか湧かない人が多いかと思います。予測値全体の確からしさを視覚的にとらえるために、検証データに対しての予測値の分布を作成します（**図3**）。実際に予測したい値を持つグループとそうでない値を持つグループに分け、それらグループに対しての予測値それぞれを分布として表現したものを作成します。この予測値の分布から、どのグループの人を高い確度で分類できているかが判断できます。

予測理由を見える化

また全体の特徴でなく、単一の予測値の場合には、その予測値がどの特徴量に重きを置いて算出されているのかを表現するのも有効です（**図4**）。こういった情報を出すことで、単純に予測値だけを出すのと比較して、実際の行動につながりやすくなります。突然あなたが機械学習のシステムから、「このお客さんが成約してくれる確率は80%！　だからアプローチしてください」と言われても納得できないですし、行動に移し辛いと思います。「このお客さんが成約してくれる確率は80%です、なぜなら去年の売上の成長率が高く、類似商品の導入実績が

注1） ある業務・ある分野に特化した知識。

▼図4　予測のもとになった特徴量を可視化（再入院する・しないと判定された理由を表示）



まだなく、キーパーソンへのアプローチも完了済みだからです」と確率を導き出した理由が示された場合には、信頼性も上がりますし、行動にも移しやすいかと思います。もちろんこの理由がしっくり来るか来ないかは、データを振り返ったり、ドメイン知識とひも付けたりしたあとの話にはなりますが、元の確率値だけよりはずいぶん安心できるものになったかと思います。

◆ ◆ ◆

機械学習を信頼するに足るものにするためには、さまざまなプロセスをしっかりと押さえる必要があります。機械学習の結果をグレーボックスにすることによって、サービス化する前の最後の予防線として使用できます。「DataRobot」にもさまざまなグレーボックス化を行う機能を設けていますので、もし製品自体にも興味がある方はぜひ筆者たちへお問い合わせください。

これから機械学習とどう向かい合うべきか

作る側は、これまで書いてきたさまざまな注意点や解決方法と向かい合う必要があります。機械学習の結果が暴走して怖いという意見もあるかもしれませんが、機械学習ではあくまで確率値が出るだけですので、それを自動化するかはそのサービスを作る側に委ねられます。機械学習によって自動化しているサービスはこれまでもいくつかありますが、使う側には裏側のロジックは見えていません。ですので機械学習が搭載されたと言っても、利用者側からは精度が良くなったという感覚しかないかもしれません。

たとえば与信スコアの判定で使われていたとした場合、与信に落ちるか受かるかという結果の種類に変化はありません。データ収集のために、手書きの手続き書類がWebサイトでの作成になるといった変化はあるかもしれませんが、受けるサービスは変わっていないものです。もちろん精度が上がって、これまでギリギリ受かっていた人が落ちたり、ギリギリ落ちていた人が受かったりと、個人個人で影響は出るかもしれません。

今後しばらくはコスト減や精度向上などの変革が大半を占めるかと思いますが、だんだんと今までは使えなかった人も機械学習を使えるようになってくると、革新的な事例がどんどん自然に生まれてくると思います。

一方でEUでは、AIが与信などの意思決定をしたときには、利用者がその理由の開示を求める権利を法制度化しようという動きもあります。利用者がより安心して新しいテクノロジの恩恵を享受できるしくみ作りも、加速していく必要があります。

今回グレーボックス化について書きましたが、大阪ガスでデータ分析に関わる河本薫さんが「データサイエンティストフォーラム2017」で話していた、「ブラックボックスを恐れている場合じゃない」という新しい視点の考え方があります。

10年経ったらブラックボックスと呼ばれていたものも普通になります。ただ、競争に勝つには10年も待っていられません。まさに今は機械学習の過渡期であって、機械学習がブラックボックスと呼ばれなくなる時代は確実に来ます。インターネットを使っている人の何割が、httpsのことがなんだがわかっているでしょうか？　ブームの中、新しい技術を使って勝つためには、ブラックボックスに飛び込む勇気を持つことも選択肢の1つだと思います。SD

人狼知能で学ぶAIプログラミング

狩野 芳伸、大槻 恭士、園田 亜斗夢、中田 洋平、箕輪 峻、鳥海 不二夫 著／人狼知能プロジェクト 監修
B5変形判／416ページ
3,680円＋税
マイナビ出版
ISBN＝978-4-8399-6058-2

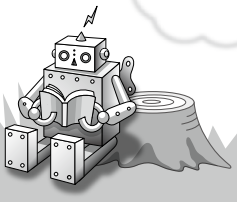
「人狼」とはテーブルトークRPGの1つで、村人や狩人、人狼といった役職を与えられたプレーヤが会話を通じた心理戦を行いながら、お互いの正体を推理していくゲーム。この人狼をプレイするプレーヤを人工知能として実装したのが「人狼知能エージェント」で、エージェント同士をWebの掲示板で戦わせる大会が2015年から毎年開かれている。本書では、このエージェントを作るためのSDK「人狼知能プラットフォーム」を使った開発手法、サポートベクターマシンにフォーカスした機械学習、形態素解析・品詞解析といった自然言語処理の分野を解説している。本プロジェクトでは、最終的に人間と戦うに足るエージェントを開発することが念頭に置かれており、実現すれば人間の嘘を簡単に見抜く人工知能が誕生してしまうかもしれない。

動かして学ぶセキュリティ入門講座

岩井 博樹 著
A5判／264ページ
2,200円＋税
SBクリエイティブ
ISBN＝978-4-7973-8746-9

本書は全体が3つのパートに分かれていて、Part1はセキュリティの現状把握、Part2はキーワードの解説、Part3はWindows上でのセキュリティ対策について書かれている。Part2のキーワード解説部分は、混同しそうなセキュリティ関連のキーワードについて図を使って説明されているので、一通り目を通して自分がどのくらい理解できているかをチェックすることができる。Part3以降は、Windowsで最低限行っておくべき設定（Windows上でのファイアウォールやWindows Defenderの設定方法）や、攻撃別の対策方法やチェックのしかた（この辺りが動かして学ぶ部分）、いくつかのツールの導入について解説されている。基本的なセキュリティについてまんべんなく学習するのに適しているだろう。

SD BOOK REVIEW



プログラマのためのGoogle Cloud Platform入門

阿佐 志保、中井 悦司 著
B5変形判／296ページ
3,000円＋税
翔泳社
ISBN＝978-4-79813-714-8

Google Cloud Platform（GCP）は、Googleが提供するGmailやYouTubeのインフラとベースを同じにしているということもあって、注目度が高い。本書はそんなGCPについて、実際のアプリ開発に沿ってサービスの機能・使い方を解説していく。まずは、GCPの基本やストレージ系のサービスについて紹介しながら、Webの掲示板アプリの実行基盤を開発していく。さらにこの掲示板アプリを本番公開することを想定して、ロードバランサやDNSのサービスの使い方を説明していく。基本的なWebアプリだけではなく、オンライン五目並べゲームのコンテナ実行環境や、画像認識機能を搭載した写真アルバムサービスなど高度な例もある。GCPでできることを概観しながら、チュートリアル的に学ぶことができる1冊だ。

Webフロントエンドハイパフォーマンスチューニング

久保田 光則 著
A5判／352ページ
2,680円＋税
技術評論社
ISBN＝978-4-7741-8967-3

モバイルによるWebブラウズ時間の増加、Single Page Applicationに代表されるWebページの複雑化などを背景に、Webフロントエンド高速化の重要性はますます高まっている。本書ではフロントエンド高速化について、ブラウザのしくみから、パフォーマンス計測と対策までを紹介している。ブラックボックスと思われがちなブラウザの動作、パフォーマンスの計測方法や計測指標からしっかりと解説してあるため、場当たり的ではない根本的な高速化に取り組める安心感がある。リソース読み込みに代表される鉄板テクニックから、認知的なパフォーマンス向上策まで収録テクニックが多岐に渡るのも特徴だ。Webサイト、Webアプリケーションの速度に課題を感じている人は、一読すると得るものがあるはず。

特集 2 プレゼンテーション技術力の高め方

エンジニアのためのうけるプレゼン・すべるプレゼン

――あなたの思いを伝える技術、教えます

Author 横田 真俊（よこた まさとし）　URL http://wslash.com/
さくらインターネット㈱　エバンジェリスト

誰でもプレゼンは怖いものです。エンジニアにとって自分の拠り所となるのは「技術力」ですが、その発展と進歩はいつでも目覚ましく、それだけでは自分を引き立てられなくなってきています。そこで、発表する力＝プレゼン力の出番です。今まさにITエンジニア勉強会ブームです。毎日のように勉強会が開催され、さまざまなテーマでプレゼンが行われています。プレゼンで人生が変わった人も少なくありません。本特集では、自分の人生を前向きにコントロールする技術として、プレゼンの達人にその極意を執筆していただきました。これを手がかりに改善してみませんか！　今後のプレゼンがうけるか、すべるかはあなた次第！

イラスト：高野涼香

特集 2

プレゼンテーション技術力の高め方

エンジニアのためのうけるプレゼン・すべるプレゼン

――あなたの思いを伝える技術、教えます

Author 横田 真俊（よこた まさとし） http://wslash.com/ さくらインターネット㈱ エバンジェリスト

Lesson 1 意識改革 編 ―― なぜエンジニアもプレゼンができたほうが良いのか?

1st Stage 広がる「エンジニア」の職務

あなたはプレゼンテーションが得意ですか？

もしあなたがエンジニアであれば、人前に出て話すことは苦手と感じている方も多いでしょうし、そういうプレゼンみたいな仕事は営業職や企画職に任せておけば良いと考えている方も多いでしょう。

しかし、エンジニアの職務として、

「プレゼンテーション」の能力は
本当にいらないのでしょうか？

筆者はエンジニアの方にこそ、

「プレゼンテーション」の能力は
必要になってきている

と考えています。

たとえば、あなたがWebサービスのサービス運営などを任せられたとします。あなたはエンジニアとして、そのサービスの作成やインフラ周りの整備を行います。ただ、あなたの仕事はエンジニアリングだけではありません。サービスの運営全般を任せられた場合、エンジニアリング以外にもさまざまな仕事が出てきます。

まず、上司に自分の企画を説明する必要がありますし、場合によっては他部門の人達や、そのサービスを売ってくれる営業の方にも説明が必要となるでしょう。

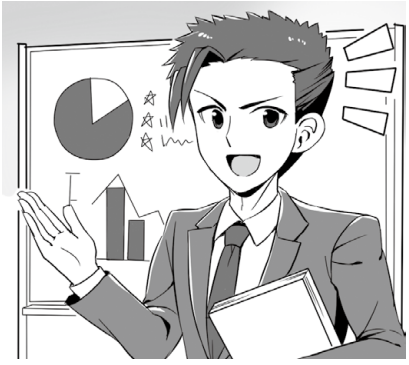
さらに場合によっては、

自社製品のアピールのためにお客様の
前で話をして、競合とのコンペに
勝たなければなりません。

このようなエンジニアリング以外の仕事をする場合に、

大きな武器となるのが
「プレゼンテーション」の能力です。

自分が作成しているサービスがどのような物であるかを上司やお客様に説明する必要も出てきますし、場合によっては大勢の前で自分が担当しているサービスの説明をする場合も出てくるかもしれません。そのようなときにうまくプレゼンができれば、自分が担当したサービスの



評価が良くなるでしょう(もちろん、あなた自身の評価もです)。

2nd Stage 自分の仕事の説明を「他人」任せにしていませんか?

ここまで読んで「プレゼンや客先説明は営業や企画の仕事だろう、なぜ我々がプレゼンをしなければならないのだ?」という方もいらっしゃるかと思います。確かにお客様に商品説明をするのは、営業の方がすることが多いでしょうし、社内調整などは企画の方がすることもあるでしょう。しかし、

それでも筆者はエンジニアが自分で上司やお客様に説明する機会を増やしたほうが良い

と思います。

たとえば、自分が人数が少ないベンチャー企業に就職した場合、お客様への説明などについて、専任の人がいるとは限りません。自分でお客様に自分のサービスを説明する場合も出てきます。そのとき「プレゼン」に慣れていなければ、お客様にうまく説明ができないかもしれません。

また、営業や企画職などが専任でいる場合でも「プレゼン」ができるようになったほうがメリットがあります。エンジニアとしてサービスを作るときに、自分が手がけるサービスについて、第三者が説明をして相手に正確な意図が伝わらない場合もあります。そのようなときに、はじめから自分でサービスについて説明をしたほうが良いとは思いませんか?

サービスについて一番わかっているのは自分なので、第三者を介さずに話したほうが正確に意図を伝えられるでしょう。もし「人前で話すことが苦手だから、プレゼンは他人にしてもらう」という気持ちであれば、

自分のサービスや企画を相手に直接伝えるというすばらしいチャンスを捨てている

ことになっていると思います。最近では技術的な事柄を普通の人に説明する、

「エバンジェリスト」

という職種もIT業界を中心に増えてきました。営業職や企画職に比べてエバンジェリストは、言わばプレゼンのプロです。このエバンジェリストのような存在がいれば、エンジニアであるみなさんが、わざわざプレゼンをしなくても良い気もします。

エバンジェリストという職種は、まだ世間一般ではそれほど知れわたっていないため、

人によっても定義は違いますが、一般には自社のサービス・技術に関係することを世間一般にわかりやすく説明する人達で、プレゼンや記事執筆を通常のエンジニアや営業職よりも多く担当している場合が多いでしょうし、数を多くやっている分プレゼンもうまいはずです。このようなエバンジェリストのような職種があるのですから、プレゼンはそのような職種の方にお任せすれば良いと思う方もいらっしゃるかもしれません。

確かにエバンジェリストのような人が会社にいれば、その方に話してもらえば自分は人前に出て話をしなくとも良いでしょう。ただ、

すべての会社にエバンジェリストのような「プレゼンが得意な人」がいるとは限りませんし、自分が企画・作成したサービスを自分の意図どおりに説明してくれるかはわかりません。

さらに言えば、そのようなエバンジェリストには、たくさんのプレゼンの依頼が来るでしょうから、自分の都合の良いときにプレゼンをしてくれるとも限りません。

そして会社と関係ない、自分のサービスについてはどうでしょうか？　自分の趣味で作成したサービスについて、自分以外でそれを説明してくれる人はいないはずです。このような場合は、自分でそのサービスを説明するしかありません。結局のところ、

「エバンジェリストがいるから自分が説明しなくても良い」ということにならず、自分で説明する必要

が出てきます。

3rd Stage 社内外に認められるために「プレゼン」を武器として使いましょう!

仕事以外でも「プレゼン」を活用することで社外からの評価が上がることがあります。最近では数多くのエンジニア向けの勉強会やセミナーが開催され、エンジニアの方が発表する機会も増えています。以前は「セミナーで登壇」というと選ばれた人が大人数の前で発表するという印象がありましたが、最近では大きなセミナーはもちろんのこと、小さい規模の勉強会も数多く開催されるようになり、

登壇することのハードルは非常に低くなりました。

もし、あなたがエンジニア向けの勉強会に参加したことがあれば、有名なエンジニアの方以外の人、たとえば自分の知り合いが登壇しているところを、見たことがある方も多いでしょう。それだけ登壇は普通のことになってきています。

このように、エンジニア向けの勉強会が増えたことで登壇する機会は増えましたが、それではこのような勉強会に登壇することは、どのようなメリットがあるのでしょうか？　金銭的な面では、登壇すると謝礼が出るような大きなセミナーはともかく、小規模な仲間内の勉強会にメリットはない気がします。しかし、勉強会で登壇すれば、その勉強会での知り合いも増えますし(少なくとも登壇することで、参加している人はあなたの顔と名前を見てくれます)、

そこから自分の関心のあるテーマについて仲間が増えたり、人脈が広がったりするかもしれません。

実際に、勉強会の登壇をきっかけにして転職につながった人もいます。たとえば、あるクラウドベンダの勉強会では、勉強会で良いプレゼンテーションをした人が、そのクラウドベンダから声をかけられ、そのベンダに入社したこともありましたし、勉強会やセミナーの登壇をきっ

かけにして転職先が見つかる方も増えています。もちろん勉強会で登壇することの目的は「**転職**」だけではありませんが、

プレゼンによって新しい職を
獲得することも可能

なのです。

また、このような勉強会の発表をきっかけに、自分が作成したサービスの知名度を上げられるかもしれません。セミナーや勉強会での発表をきっかけにして注目を集めたサービスもいくつもあります(もちろん、その場合はプレゼン能力だけでなく、ソフトウェアやサービスの「出来」がよくなければいけませんが……)。

このように「エンジニアなのでプレゼンテーションをしない」という状況が変化してきています。むしろサービスやツールなどを作成するために、上司や同僚の同意を得るために、また自分が作成したサービスやツールを広めるために「プレゼン」を武器として利用するシーンが出てきていると言っても良いでしょう。

4th Stage プレゼンの苦手を克服して、社内外を「説得」するツールとして利用しよう!

「プレゼンの重要性はわかっている。
でも、人前でしゃべることは苦手だ」

という人もいます。どんなに技術が優れた人でも、大人数の前でしゃべることにとても苦手意識を持っている方もいます。「プレゼン」についてのよく言われている小話に、

「人前で話すことは
死ぬことよりも恐ろしい」

というものがあります。プレゼンを苦手に思う人の心を代弁したような回答ですが、プレゼンや講演会の登壇者が「プレゼンは死より恐ろしい」と言って講演会に来なかったことはありませんし、

プレゼン中に死んだ人も
あまり多くいない

と思います(もし、プレゼン中に死んでしまった人がたくさんいたら、法律でプレゼンは禁止されるでしょう)。

しかし、この「人前で話すことは死ぬことよりも恐ろしい」という小話はまことしやかに言われています。この小話はもともとは『The Book of Lists』という1977年に出た雑学本に書いてあったもので、3,000人のアメリカ人に「もっとも怖いものは何ですか?」というアンケートをとったところ、1位が「人前で話すこと」で7位が「死」だったという結果[注1]が出ました。このことから「人前で話すことは死ぬことよりも恐ろしい」という部分が広まったと言われています。

column コラム

Dockerの流行を生み出したライトニングトーク

本誌を購入されている方であればDockerについては、少なくとも聞いたことがあると思います。登場してからわずかな間に広まったDockerですが、もともとはPyconというPythonのカンファレンスの中のライトニングトークで初めて発表されたのが流行のきっかけです。Dockerは、このライトニングトークからさまざまなテック系メディアに掲載されるようになり、人気プロダクトになりました。このように現在では当たり前に利用されているDockerも最初に発表されたのは、カンファレンスのライトニングトークからです。今後、あなたが個人的に作成したツールやWebサービスをセミナーなどで発表する機会があるとき、ライトニングトークや講演でプレゼンがうまくできれば、Dockerのように人気がでるかもしれません。

注1) https://www.amazon.com/Peoples-Almanac-Presents-Book-Lists/dp/0688031838/ref=pd_sim_14_3?_encoding=UTF8&pd_rd_i=0688031838&pd_rd_r=1MJNDM4FVD417Q100KFV&pd_rd_w=PUwiW&pd_rd_wg=z0BN6&psc=1&refRID=1MJNDM4FVD417Q100KFV

人前でしゃべることについて苦手意識がある人が多いからこそ、このような小話が幅広く広まっているとも言えます。普段、仕事のため人前に出てしゃべる機会が多い人はともかく、そうでもない方が人前でしゃべるのはなかなか慣れないと思います。

実際にIT業界の有名な人でも、最初の講演は失敗したと感じている人もいます。たとえばLinus Torvaldsもその1人です。

Linus Torvaldsと言えば、みなさんもご存じのとおりLinuxカーネルの開発者です。彼の最初の講演について、彼の自伝でもある『それがぼくには楽しかったから[注2]』に、こんなふうに書かれています。

> *「講演はどうだったかって？　聴衆は、彼らの前に立ち、救命胴衣のようにパワーポイントのスライドにすがりつき(マイクロソフトよ、ありがとう)、質問にたどたどしくこたえる見るからに怯えきった若者に同情していた。(中略)初めての講演は、ショック療法みたいなものだった。その次の講演も同じようなものだったが、段々と自信がついてきた」(『それがぼくには楽しかったから』P.180より)*

あのLinus Torvaldsでさえ、最初の講演はこのように失敗して、段々と自信がついてきたと語っています。

かくいう筆者も人前で話すことは
非常に苦手でした。

初めて大勢の前で講演をしたときはネットワーク技術者が集まるイベントで、筆者はその中でプレゼンをしたのですが、講演中はあまりにも緊張してしまい、

自分でも何を発表しているのか
わからない状態

でした。どれぐらい筆者が緊張していたかと言えば、プレゼンの発表中に聴衆の中から「ガンバレー」という声援をもらったほどです。確かにこれだけ大勢の前で恥をかくのであれば、

「人前でしゃべること」は
「死ぬことより」も恐ろしい

のかもしれません。このように筆者は、当初は人前に出てしゃべることは苦手でしたが(何しろ聴衆から励ましの声援をもらっていたほどです)、

今ではエバンジェリストとして
さまざまなテーマで年間50本ほど
講演やハンズオンを実施

しています。

これは筆者が「話し方がうまかった」ということでも「スライド資料の作り方がうまかった」ということでもありません。もともと話し方もスライドの作成方法も下手でしたが、

それでもプレゼンを多く実施することで
「話慣れ」をした

ことや、

注2) リーナス・トーバルズ、デビッド・ダイヤモンド(著)、風見潤(翻訳)、中島洋(監修)、『それがぼくには楽しかったから――全世界を巻き込んだリナックス革命の真実』、小学館プロダクション、2001年

column コラム

質問タイムに「演説」をする人をどうするか?

多くのプレゼンや講演のあとには質問タイムがあります。質問がたくさん出るプレゼンは、聴衆が関心を持ってくれている証拠なので、講演者にとって質問自体はうれしいのですが、中には困った質問者の方もいらっしゃいます。筆者も最初のころは、なかなかこのような方にはうまく対応できませんでしたので、そちらの失敗談を紹介します。

ある技術論的なプレゼンのあとに、質問タイムがあったのですが、質問者の方が延々と自説を述べられ、まるまる質問の時間を食い潰してしまったことがありました。質問をしていただけるのはありがたいのですが、多くの聴衆にとって、その人の質問(というか自説)はあまり関心がないものだと思います。

本来であれば、「ほかの質問者の方もいるので……」とか「質問の時間も限られているので……」など、うまく質問を終わらせるような言葉をはさんで質問を切り上げるようにすべきでしたが、筆者はまだプレゼン自体にも慣れておらず、質問者に対してどのように接すれば良いかわからなかったので、うまく対応できませんでした。

「質問タイムに演説をする人」に対応するには、質問を切り上げるほかにどのようにしたら良いのでしょうか? 同じ種類の勉強会やセミナーに何度か行くと、なんとなく「演説をする人」がどのような人なのかわかってきます。そのような人達の質問については、質問タイムの最後の方に指名するなどの工夫をすれば、みなさん納得されると思います。

ちょっとしたことを気を付けるだけで
プレゼンの出来がよくなった

ため、これだけのプレゼンを行うことができるようになりました。

今回の特集記事では、

筆者がここ数年、気を付けてきた
「プレゼン」が苦手ではなくなる
コツについて説明をしていきます。

みなさんのプレゼン技術向上のお役に立てれば幸いです。

Lesson 2 実践入門 編 —— プレゼンの「練習」をする場を作る

1st Stage 「プレゼンのコツ」を書いた本や記事を読むだけでは上達しません!

プレゼンやスピーチについて書かれた本や雑誌の記事は山のようにあります(まさにこの記事もそうですが)。最近ではプレゼンに関するWeb記事などもたくさんあるので、検索エンジンで「プレゼン　やり方」「プレゼン　うまくやる方法」などのキーワードで検索をすれば山のように記事が見つかるでしょう(**図1**)。

しかし、これらの記事をいくら読んでも、ただ読むだけではプレゼンはうまくならないでしょう(それは、今あなたが読んでいる／筆者が書いているこの記事も含めてです)。確かに、これらのプレゼンに関する記事はいろいろと勉強になることもあるかとは思いますし、スライドの作り方などの記事は、スライド作成時に参考になるかとは思います。

▼図1　キーワード検索画面

しかし、初めてプレゼンをする人が、

「話し方」や「プレゼン技術」に関しての
本や記事を読んで、
すぐにうまいプレゼンをする

のはなかなか難しいでしょう。うまいプレゼンをするには、これらの本に書かれていることを実際にやってみて経験を積む必要があります。

「プレゼンの本を読めばプレゼンがうまくなる」
というのは「野球やサッカーの本を読めば
野球やサッカーがうまくなる」

と言っているようなものです。野球やサッカーの本には、トレーニングの方法などは書かれていると思いますが、読んだだけでうまくなると思っている人はいないでしょう。

プレゼンも同様に単に記事や本を
読んだだけでは身につきません。

うけるプレゼン　本を参考にしつつ、実際にプレゼンをこなしている

すべるプレゼン　本を読んでできるつもりになってしまう

2nd Stage プレゼンがうまくなるためには場数をこなすのが一番

では、どのようにすればプレゼンはうまくなるのでしょうか？　筆者は、

実際にプレゼンの場数を
こなすことが一番

と主張したいと思います。プレゼンに関する記事にはとても良いことが書いてありますが、実際に人前に出てプレゼンをするときは(とくに初めて人前に出てしゃべる人はなおさら)プレゼンで記事や本に書いてあったとおりに話せる人はそんなにいないでしょう。

ただ、最初の1～2回は失敗しても、何回もプレゼンをしていると、

最初はできなかったことができるように
なってきますし、場慣れしてくると
プレゼン中のトラブルにも
対応できるようになってきます。

まずはプレゼンをする機会を増やして実際に人前に出てしゃべってみるのが上達への近道です。何度も人前でしゃべれば、

「人前で話をすること」にも慣れますし、
話をすることに慣れると
プレゼンの内容に自信が出てきます。

そして自信が出てくれば、聴衆もより熱心に話を聞いてくれるようになるでしょう。

プレゼンがうまくなる王道は、
実際にプレゼンの数を増やして
プレゼンの場数を増やすことです。

プレゼンの場数を増やすことで、だんだんとプレゼンがうまくなっていきます。

うけるプレゼン　とにかくプレゼンの場数をこなす

すべるプレゼン　プレゼンを経験した回数が少ない

3rd Stage「練習」と「失敗」を繰り返せば、プレゼンはうまくなる

プレゼンの場数を増やせば確かにうまくなります。ただ、もしこの記事を読んでいるあなたが1週間後に大事なプレゼンがあった場合、どうでしょう？　たぶん「プレゼンは場数をこなせばうまくなる」というアドバイスはあまり役に立たないかもしれません(何しろプレゼンは1週間後です)。

もし、あなたが近いうちに重要なプレゼンを実施するとき、

そのプレゼンを成功させたいと
思っているのであれば、
そのプレゼンの「練習」をすべきです。

不思議なことにプレゼンを成功させたいと思っている人はスライドにギリギリまで手を入れることには熱心ですが、実際に、

自分が作ったスライドを見ながら
リハーサルや練習をすることは
あまりやっていない

ように思えます。プレゼンで使うスライドを早めに完成させたら、

自分のプレゼンのリハーサルを
してみましょう。

一度リハーサルをしてみると、自分のプレゼンスライドや自分の話し方の問題が見えてくるかもしれません。その部分を修正して再度練習をすれば、最初に行ったリハーサルよりも、もっとうまくできるようになっているでしょう。これを繰り返せば、

そのプレゼンが成功する確率は
高まっていくでしょう。

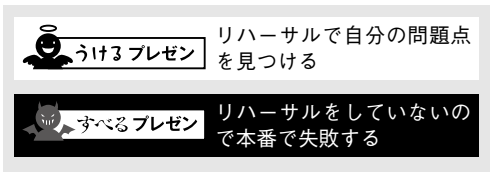
プレゼンを実施するときは、どうしてもスライドをギリギリまで作成してしまい、プレゼンのリハーサルや練習をしない人はかなり多いと思います。中にはプレゼンに練習など必要ないと思っている人もいるかもしれません。

確かに、練習に付き合ってくれる人がいなければ、誰もいない空間の中1人でしゃべるのはバカみたいですし時間もかかります。しかし、

リハーサルを行えば本番前に問題点を
見つけられますし、本番のプレゼンの
前に「失敗」することもできます。

本番では失敗は許されませんが、自分で勝手にやっているリハーサルでは「失敗」しても問題にはなりません。

もし、重要なプレゼンが迫っているのであれば「練習」と「失敗」を繰り返しておけば、プレゼン本番で失敗することはないでしょう。



4th Stage プレゼンのチャンスを積極的に作っていますか?

プレゼンがうまくなるのは「場数をこなすこと」や「リハーサルや練習をすること」だと説明をいたしましたが、実際に「場数をこなす」と言っても、

登壇依頼が来ることは、多くはない

と思いますし「リハーサルや練習」と言っても、直近で自分が発表することがなければ、とくに「リハーサルや練習」などをしないでしょう。

もし、プレゼンをする場所がなければ、

自分でプレゼンをする場所を探し

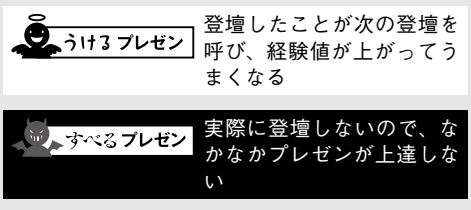
て、そこでプレゼンをする機会を作ってみましょう。自分が属している会社で勉強会などがあれば、そこでプレゼンをすれば良いでしょうし、自分がよく行く勉強会などで発表をする機会があれば、そこで発表すれば良いでしょう。

「発表する機会」を増やしていけば、それ自体が「発表する練習」になります

し、何回も登壇することで、人前で話すことに慣れてきます。

「場数をこなす」

ということがプレゼンの上達の近道ですが、登壇依頼を待っていないで、自分で発表する機会を作ってみましょう。



5th Stage 勉強会やLTで話してみましょう!

しかし、自分の周りで自分が発表する機会がない場合は、どうしたら良いでしょうか? 「プレゼンの場数をこなせばうまくなる!」と言わ

column コラム プレゼンのリハーサルについて

実際にはどのようにリハーサルをすれば良いのでしょうか? リハーサルのやり方はいろいろとあると思うのですが、ここでは筆者がやっているリハーサルの例を紹介します。

まずは完成したスライド資料を実際にプレゼンテーションモードで一通り見てみましょう。自分で作ったスライドですが、通して見ると順番がおかしかったり、誤字脱字が見つかったり、スライドのアニメーションがおかしい個所があるものです。一度スライドショーモードで確認をすれば、このようなミスはだいぶ減らせると思います。

次に、本番のように声を出してスライドを発表してみましょう。指定された発表時間を過ぎないように、タイマーや時計で時間を計りながら実施してみましょう。時間が足らないようでしたら、スライドの説明を簡略化するか、思い切ってスライドを削ってみましょう。

また、仕事の重要なプレゼンの場合は、社内のチームのほかのメンバーに練習に付き合ってもらうことをお勧めします。とくに会社の重要な製品発表や社内でも大きなイベントの場合は、複数人で自分の発表をレビューしてもらうべきです。

筆者も自分が担当しているクラウドサービスの製品発表会や、年末に実施している会社の大きなイベント用の発表では同僚や上司とリハーサルを行って発表いたしました。さすがに通常のイベントでは、複数人でのチェックはいらないと思いますが、ここ一番では第三者も入れてリハーサルをしたほうが良いでしょう。

れても、そもそも人前であまり話しをしない方からすると、

「人前でしゃべる機会なんてない」

と思うかもしれません。そういう場合は、発表できる場所を見つけてみましょう。最近ではエンジニア向けの勉強会は毎日のように開催されています。その勉強会の中には、

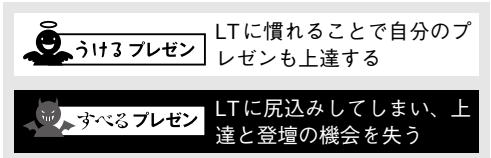
登壇者やLT（ライトニングトーク：Lightning Talk）を募集

しているものもあります。その勉強会の講演枠やLTでしゃべってみてはどうでしょうか？

「登壇をする」と言うと大げさに聞こえるかもしれませんが、LTは基本的に3分から5分程度の短い発表をするもので、

通常のセミナー講演に比べれば、ハードルが高くありません。

どのような勉強会のLTに参加すれば良いかわからない方は、まず、自分がよく行く勉強会や関心があるテーマを取り扱っている勉強会のLTに参加すると良いでしょう。なじみのある勉強会ならば、そこで聞いてもらえるテーマがイメージしやすく、発表するテーマも見つけやすいでしょう。勉強会によっては、LTだけを取り扱う勉強会や、初めてLTする人を優遇する勉強会もありますので、初めてLTに登壇をする人はそのような勉強会を探してみるのも良いでしょう。



6th Stage 「失敗」をして経験を積むことが大事

LTに応募して、

スライドを作るときや、

発表をするときは、いろいろと後悔することも多い

と思います。なぜかと言えば、スライドを作るときは、発表時の締め切りに追われることになりますし、LTをするときは最初は緊張してうまく話すことができないかもしれません。また、スライドも慣れている人に比べたら効果的に使えないかもしれません。話をしているときは、

緊張で汗だくになる

こともあるでしょう。最初のLTはいろいろ失敗してしまうかもしれませんが、

最初はそれで良い

のです。LTなど人前で話すことを繰り返していけば、

人前で「話慣れて」きます。

そうすると自分なりにプレゼンのやり方がわかってきます。実体験で得られた経験は、プレゼンのコツを書いた本やWeb記事をいくら読んでも身につきません。

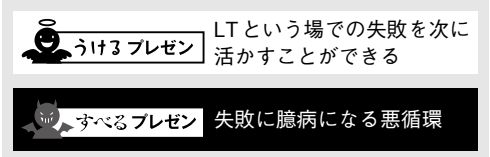
LTなどをするときは、最初は「**失敗**」するかもしれませんが、ここで失敗しても、多少の人間の前で恥をかくだけです。しかし、仕事で行うプレゼンなど「**失敗できない**」ときがあると思います、そのときに「**失敗**」するよりも、

「失敗しても良い」ときに失敗

して、

プレゼンの経験値を貯めたほうが良い

のです。



column コラム

LTで場数をこなしてうまくなった

ここまで「プレゼンは場数をこなせばうまくなる」と説明してきましたが、これは筆者以外の人達にも当てはまります。実際に何回かプレゼンを行っている人達は確実にうまくなっています。

後述しますが、筆者と当社同僚で一緒に「プレゼン研究会」という勉強会を主催しています。この勉強会では、参加者全員に1分間の自己紹介とテーマに沿ったLTをやっていただきます。

参加者全員が事前にスライドを作り、しかも実際にプレゼンをやるというハードルが高い勉強会です。例外はなく、スライド資料を作り忘れた人がいれば、その人がスライド資料を作り終えて発表ができる状態になるまで、ほかの人が何回かプレゼンで場をつないだこともありました。

このようなハードルが高い勉強会ですが、何回か参加されている人達は確実にうまくなっています。勉強会初回参加時にスライド資料を作り忘れた方も、初回は即席で作った資料だったので満足できるクオリティではありませんでしたが、何回か実際にプレゼンを行った結果とても上達されました。

7th Stage 勉強会を主催していますか？

さまざまな勉強会でLT登壇を経験してきたら、今度はLTではなくもう少し長い時間話をするようなプレゼンをしてみましょう。もし、何度かLTをやったのが縁で勉強会の登壇依頼が来たら、ぜひ受けて登壇してみましょう。スライド資料を作るのも、話を長時間するのもつらいと思いますが、LTをするよりもいろいろな経験ができます。

ただ、LTと違い長時間の登壇については、なかなかチャンスがこないかもしれません。

そんなときは自分達で勉強会を主催してみるのはどうでしょうか？

自分で主催する勉強会であれば、自分の好きなテーマで話をすることができますし、しゃべる時間や一緒に登壇する人も自分達で決められます。

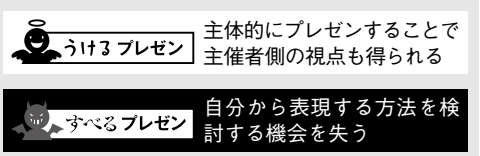
たとえば、筆者は同じ会社のテクノロジエバンジェリストである前佛雅人氏と一緒にプレゼンに関する勉強会「プレゼン技術研究会」というイベントを定期的に開催しています。この勉強会の特徴は、参加者全員がテーマに沿ってプレゼンをするということです。参加者全員がプレゼンをすることになるので、事前準備もたいへんですし、参加者のプレゼンを全部見る必要があります。

全員でプレゼンとなると、参加側にも主催者側にもかなりの負担となるため、通常の勉強会やセミナーでは実施しないと思いますが、自分が主催者となれば、このような形態の勉強会もできます。

実際に勉強会やセミナーを開催する場合、会場の手配などの問題もありますが、仲間内のクローズドな勉強会であれば告知を出す必要はありませんし、会場も会社の会議室や喫茶店の個室や会議室を利用すると良いです。

自分達で勉強会を主催すれば登壇するテーマを選ぶことができますし、自分が話したい、もしくは聞きたいテーマで勉強会を設定できます。

人の手配などたいへんですが、機会があれば自分で勉強会を主催しても良いでしょう。



column コラム

勉強会を開催するときの人数カウントに注意

クローズドな勉強会ではなくオープンな勉強会を開催するときは「connpass」や「Doorkeeper」などのサービスを利用して集客をすると思います。開催告知を出したら意外に人が集まるかもしれません。

しかし、申し込みをしたすべての人達が来るわけではありません。さまざまな勉強会がカジュアルに開催されるようになり、キャンセルするのもカジュアルにできるようになってしまい、最近の勉強会は非常にキャンセル率が高いです。

筆者も「プレゼン研究会」に限らずいろいろな勉強会の企画をしていますが、登録者の割には参加している人間が大幅に少ないこともあり、会場選定や食事の用意に苦慮したことがあります。

これは筆者の感覚なのですが、イベントに参加登録をした人のうち実際にイベントに来るのはよくて登録者の8割、悪ければ半分も来ないでしょう。

ですので、もし想定より参加者が多く参加したとしても、あわててもっと大きな会場を用意する必要はありません。参加者は多くて8割ぐらいの感覚でいたほうが良いと思います。

Lesson 3 即効技術 編 —— 聴衆を意識してテーマを作る

1st Stage 本当に訴求したいテーマは何ですか?

プレゼンは「数をこなせばうまくなる」と言われても、

話すテーマが決まっていなければ
プレゼンをすることはできません。

自分が登壇する場合のテーマは、どのようにすれば良いのでしょうか？　依頼を受けてプレゼンをするのであれば、プレゼンをするテーマは決まっていますが、LTや自分が主催の勉強会については自分でテーマを設定しなければいけません。自分でプレゼンのテーマを考えるときは、どのようなテーマにすれば良いのでしょうか？　まずLTなどのように、テーマが比較的自由に選択できる場合は、

自分が訴えたいテーマで
プレゼンを作れば良い

でしょう。自分が作ったWebサービスや自分の好きなツール・言語など、自分が訴えたいことやテーマでプレゼンをすれば良いでしょう。もし「訴えたいテーマ」がない場合は、どうすれば良いでしょうか？　そのときは自分が得意な物や普段使っているツールなどをプレゼンすれば良いでしょう。自分が利用しているツールのメリットや注意点、まだ出始めのツールやWebサービスなどの利用体験などを話せば良いのです。

たとえば、「出始めのツール」であれば、各社のクラウド事業者の新サービスや、動きが速いコンテナ系や構成管理ツールなどを取り上げるのが定番です。また、それではありきたり過ぎるという方は、普通の人があまり見ないようなツールの紹介でも良いでしょう。

またLTの場合、勉強会のテーマと自分の趣味を強引に結びつけて発表するという方もいらっしゃいます。ITインフラエンジニアのみなさんの中には、アニメが好きな方も多いせいか勉強会のテーマとアニメを結びつけて発表される方もいます。

記憶に残ったものとしては、シェルスクリプト関連の勉強会でのLTでした。シェルスクリプトを使って青空文庫から「走れメロス」を抜き出し、その登場人物を特定のアニメの登場人物に変換するというプレゼンをされる方がいらっしゃいました。

そのほか、実際の上場企業役員の方をゲスト

に迎えて「自分のアイデアを発表して上司から承認を得る方法」をテーマにした勉強会を開催したときは、当時非常に人気があったアニメで、その舞台となっている荒廃した場所をいかに再建するかというプレゼンをされた方もいらっしゃいました。

今回はアニメを例に出してみましたが、別にそれ以外でもLTぐらいの短いプレゼンであれば、みなさんが持っている趣味と勉強会のテーマを結びつけて発表してもいいでしょう。

うけるプレゼン　話すテーマが決まっているから伝わる

すべるプレゼン　テーマがないから発表の軸がぶれる

2nd Stage 誰が自分の話を聞いているか意識していますか?

テーマが決まれば、あとはスライドや資料の作成をするのですが、テーマの選定や資料作りで大事なことがあります。それは、

「誰が自分の話を聞いているか？」

ということです。プレゼンをするのであれば、聞いている人に関心を持ってもらえるテーマを選ばなければなりません。LTなど登壇時間が短く、テーマが自由に決められるのであれば、そこまで意識をしなくとも良いと思いますが、登壇を依頼されたり、長いプレゼンをするのであれば、

誰が自分の話を聞いているかを
意識してプレゼンのテーマを作成

したほうが良いです。たとえば、あなたが、あるクラウドサービスについてプレゼンをするとしましょう。あなたは普段利用している、ある「クラウドサービス」についてのすばらしさをプレゼンしようとするとします。

もし、プレゼンを聞く人がすでにその「クラウドサービス」についてよく知っている人達であれば、そのサービスについての概要を説明し

ても、すでに知っていることなので関心を持ってもらえないかもしれません。反対にその「クラウドサービス」について、知らない人がほとんどであればサービスの概要などを説明したほうが良いでしょう。

せっかく自分が登壇をするのですから、

自分のプレゼンに関心を持って
もらわなければなりません。

そのためには「誰が自分の話を聞いているか？」ということを意識して資料作りや話し方を考えたほうが良いです。そうしなければ、聴衆はそのプレゼンに関心を持ってもらえないでしょう。

うけるプレゼン　誰が聞いてくれるのかわかっている

すべるプレゼン　自分が興味のあることだけしか表現できない

3rd Stage 聴衆がどのような人達か調べていますか?

聴衆のことを考えてスライドや資料を作るときは、自分のプレゼンの参加者がどのような人達なのか知っておいたほうが良いでしょう。聴衆がどのような人なのかがわかれば、その会場にあったプレゼンを用意できます。もし可能であれば、

自分が登壇するイベントにどのような人達が参加するか調べておきましょう

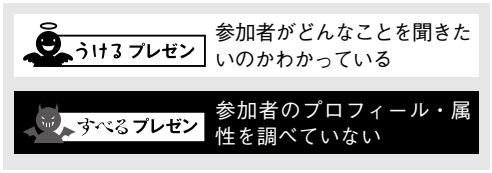
何度か開催された勉強会であれば、過去の勉強会の参加者や発表テーマを確認すれば、その勉強会の会場には、どのような人が来ているのかだいたい推測できるので、どのような発表に関心を持ってもらえるのか推測できます。さらに可能であれば勉強会の主催者に、

「会場に来るのは、どのような人達なのか？」

また、

「今回の勉強会がターゲットとしているのは、どのような人達なのか？」

といったことを確認しておくと、その会場にあったテーマを設定しやすくなります。LTなどは、自分が好きなテーマで発表できますが、それでも発表する会場に合うようにアレンジしたほうが自分のプレゼンに関心を持ってもらえます。



4th Stage なぜプレゼンで自己紹介するのか、理由をご存じですか?

「関心を持ってもらう」ことでは、

プレゼンの最初に行う自己紹介も重要

です。どのようなプレゼンでも、最初に登壇者の自己紹介や所属している会社説明が入っています。なぜ、最初に登壇者の自己紹介や所属している会社の説明をするのでしょうか？　それは、

「プレゼンをする登壇者が、どうしてこの場にいるのか」知ってもらうため

に必要だからです。自己紹介や会社説明を通じて、なぜ登壇者がそのテーマについて話をするのか、そのような登壇者についての背景を聴衆に説明するためです。

具体的に筆者が「プレゼン勉強会」のプレゼンで利用している自己紹介の資料を紹介します(**図2**)。ここでの自己紹介の資料は、筆者が会社の「エバンジェリスト」であることと「年に50回ほどプレゼンをしている」という2点に絞って紹介しています(**図3**)。

これは「プレゼンの勉強会」というプレゼンテーションの勉強会で話をするのですから、それなりにプレゼンをよくやっている「エバンジェリスト」であることや、この記事でも書いてあるとおり「プレゼンがうまくなるのは場数をこなす」ということを訴えるために「年に50回」という数を出しています。

ただ「年に50回」というのは、講演の回数として多いのか少ないのか、よくわからないと思いますので、補足として2枚目に「年に50回な

▼図2　自己紹介で所属を説明

▼図3　さらに年間プレゼン回数を紹介

ので1週間に1回はプレゼンをしています。」と説明しています。正直なところ講演が多いエバンジェリストであれば年に50回というのは、それほど多くの数ではありませんが、普通の人よりは多い数だと思います。

この自己紹介の例は「プレゼン」をテーマにしたスライドの例でしたが、会社の製品紹介の場合は、会社のエバンジェリストであることや現在でも開発チームにいることを説明するスライドにしたり、WordPressがテーマの場合であれば自分のWordPressの使用歴や過去にWordPressの本を出したことを紹介しています。

このようにみなさんも、たとえば、プレゼンのテーマがあるプログラミング言語に関するものであれば、そのプログラミング言語との関係性を自己紹介に入れたり、所属している組織と、そのプログラミング言語についてのかかわりを説明したりするでしょう。こうした説明をすることで、

「なぜ登壇者である自分がこの場にいて、そのテーマについて話すのか」

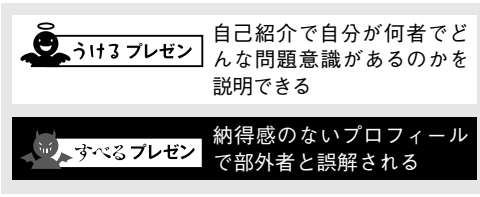
といった背景を説明できます。テーマと登壇者の背景が説明されれば、

プレゼンに対しての納得感が増します。

反対に自己紹介や会社説明のときに、これから話をするテーマとの関係性がない場合、聴衆に納得感を与えられないでしょう。たとえば、登壇者のプロフィールが、まったくプログラミング言語と接点がなければ、登壇者がなぜそのテーマで話すのか納得感が薄れます。そうなれば当然、その登壇者のプレゼンに対して関心がなくなっていくでしょう。

自己紹介で今回のテーマと自分の関係性を最初に説明しつつ、「自分がどうして、この場所にいるのか？」「なぜ聴衆のみなさんはそれを聞く必要があるのか？」ということを説明しておけば、聴衆に関心を持ってもらえるでしょう。

自分のプレゼンが聴衆のみなさんにとって、どんな意味があるのかを説明し「これから行われるプレゼンが、自分にとって意味のある話」という認識を持ってもらうと、関心を持って聞いてくれる人が増えていきます。



5th Stage 「会社説明だけのプレゼン」では聴衆に響きません!

ここまでは、自分の意思で勉強会やセミナーで登壇することを前提に説明をしてきましたが、自分の意思ではなく、会社の仕事でプレゼンをする場合もあると思います。こんなときは自分の会社の紹介や製品紹介を行うプレゼンをしがちです。ただ、残念ながら、会社の命令でイヤイヤやっているプレゼンの場合、プレゼンを聞いている聴衆も関心がないでしょうし、

プレゼンしている自分自身もつまらない

と思います。

もし、会社説明や商品説明を中心にプレゼンを行わなければならない場合は、できるだけその勉強会やセミナーのテーマにあった物を選び、聴衆に関心があるような物をテーマにしてプレゼンをしましょう。

言うまでもなく、会社紹介も最初の冒頭で軽く紹介するのであれば、それほど不自然ではありません。たとえば、筆者が会社紹介に利用しているのは**図4**のスライドです。

ただ、この会社紹介をダラダラと続けても、聴衆は退屈してしまうでしょう。残念ながら聴衆はあなたの会社の資本金やどのような名前の役員がいるかについては、まったく興味がありません。会社紹介のスライドを入れるとしても、どのような会社なのかを説明する程度で良いでしょう。

うけるプレゼン　会社の立ち位置を含めて、公正な立場をアピール

すべるプレゼン　つい営業してしまってひんしゅくをあびる

6th Stage いきなりスライドを作りはじめていませんか?

LTやセミナーでの発表が決まり、発表テーマも決まりました。これから発表まではスライドを作る必要があります。さっそく、PowerPointやkeynoteを開いて発表スライドを作ろうとしますが、

いきなりスライドを作り始めるのは止めておく

ほうが良いです。

なぜならPowerPointやkeynoteは確かに「スライドを作る」のには向いているかもしれませんが、

「スライドの中身を考える」には向いていない

からです。これらのプレゼンテーション用のソフトは、プレゼンの全体構成を作ることには向いていません。

まだ発表のテーマしか決まっておらず、スライドにどのような物を入れ込むのか決まっていない段階でPowerPointなどのスライド作成ツールを開いたとしても「どのようなスライド」を作るか、ということが決まっていないため、スライド作りがうまくいかないと思います。プレゼン資料を作るときは、PowerPointを立ち上げるのではなく、まずは、

プレゼンの全体構成を作成するのが良い

です。スライドのシナリオなどの全体構成を書き出し、それをもとにスライドを作り始めればスムーズに進められます。

▼図4　定番で使用している会社紹介

このような全体構成をまとめる場合のツールとしては、WorkFlowyのようなアウトラインプロセッサや、EvernoteやOneNoteのようなメモアプリケーションなどを使っている方もいらっしゃると思います。筆者は、これらのツールを使わずに、

テキストエディタで、どのような内容のスライドを書くか全体構成を書き、

それに合わせてスライドを作ります。いきなりスライドを作り出すのではなく、まずはこれから作るスライドの全体構成を作成してからのほうがスライドを作りやすいでしょう。

うけるプレゼン　まずテキストファイルで概略を作る

すべるプレゼン　スライドを最初から作ってしまう

7th Stage 最初に全体構成をまとめていますか?　スライド作りのコツ

スライドを作る時間も惜しいのに、

「スライドの前に全体構成を作る」

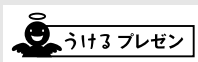
と言われても、時間のムダのように思えてしまうかもしれません。しかし、スライドを作り始める前にどのような内容を書くか、分量はどのぐらいなのか、プレゼンのシナリオをどのようにするかといった全体構成を作るべきです。

もし、何も考えずにスライドを作り始めた場合、考えながらスライドを作らなければなりませんし、どのぐらいの分量のスライドを作れば良いのかもわからなくなります。何より、スライドのシナリオが事前にできておらず、スライドを作り始めると、自分が思いついた順番でスライドを作成する形となり、プレゼンのスライドの順番がバラバラとなってしまいます。プレゼンのスライドは、

言うなれば自分のプレゼンのときの進行表

です。スライドの中身や順番がバラバラではプレゼンで主張したいことが伝わりにくくなります。プレゼンを行う人が、どんなに発表するテーマに詳しかったとしてもスライドの順番がバラバラでは、聴衆に伝わるプレゼンはできないでしょう。

最初に全体構成を書いておけば、スライドを作成する分量もわかりますし、全体構成が目次代わりになりますので、どこまでスライドを書いたのかがわかります。これらのことを事前にまとめておけば、スライドを作るときに、どのようなスライドを作るかがわかります。スライドを作り始める前に、ぜひ全体構成を書き出しておきましょう。



全体構成をしっかり作ってからスライドを作る

最初からスライドを流れで作ってしまう

◆ ◆ ◆

以前、筆者は学生向けに「コンピュータウイルス」についてのプレゼンを行ったことがあります。そのときのスライドを作る際に書き出した全体構成の一部をここで紹介します。次のように目次のような物を作成し、そこからスライドに盛り込む内容を入れてスライドを作っていきます。

```
タイトル「学生が知っておくべき昔と最新のセキュリティ」
- スマホを利用する時に知っておきたいセキュリティ事情 -

* はじめに
- コンピューターウィルスとは？
-- ウィルス、トロイの木馬、マルウェア、ワームの違い

→ まずは一般的な定義の紹介

* 実験室でのコンピュータウイルス
-- 1971年、最初期の自己複製型プログラムの一つ『クリーパー』
-- 1974年、ラビッド
-- 1981年、「コンピュータウィルス」という言葉が登場
「コンピュータウイルス」という言葉が、レオナルド・M・アドルマン（Leonard M.Adleman）教授によって、フレッド・コーヘン（Fred Cohen） との議論で使用された。

色々と「コンピュータウィルス」のような物は出てきたが、コンピュータ自体がそれほど普及していなかったので、実験室で止まっていた。

* 野生のコンピューターウィルスの登場
→ コンピュータウィルスの定義によって、どれが最初のコンピュータウイルスなのか意見がわかれる。たいていの文献では以下の二つが有力。

- 1982年、Elk Cloner（エルク・クローナ）
Richard Skrenta（当時は高校生）が開発。Apple II（アップルのMacintoshの先代に当たるパソコン）で発症。
  → 単にメッセージが表示されるだけ。
  → 元々は友人向けのいたずら目的

- 1986年、Brain
パキスタン人のアルビ兄弟が開発。IBM-PC（現在のWindowsパソコンの先代）で発症。
→ 起動時にメッセージが出るものの、特にデータ破壊などは行わない。
→ 開発の目的は、自分達の販売したソフトウェアが不正コピーされていたので、不正コピーされたソフトに
Beware of this VIRUS. Contact us for vaccination.（このウイルスに注意。ワクチンについては我々にお問い合わせを）」といったようなメッセージを出した

（以下続く）
```

8th Stage スライドの文字は大きくしていますか？　文字を少なくしていますか？

スライドを作っていると、

「あれも伝えたい、これも伝えたい」

と思ってしまい、

1枚のスライドの情報量を
多くしてしまいがち

です。実際にプレゼンに慣れていない方ですと、

1枚のスライドを小さい文字で
びっしりと埋めている方

がいらっしゃいます。ただ、残念ながら、あなたが1枚のスライドに文字をたくさん入れれば入れるほど、あなたの伝えたいことは、

聴衆には届かなくなります。

なぜ、スライドの文字が増えると、プレゼンが聴衆に届かなくなるのでしょうか？

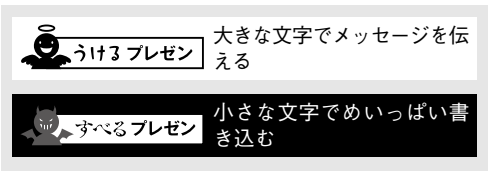
まず、単純にスライドに文字が多くなれば、

当然1文字あたりの大きさは
小さくなり読みにくくなる

ということがあります。1枚のスライドにどんなにすばらしいことが書いてあっても、プレゼンを聞いている人たちが読めないのでは意味がありません。これはスライドを作る人の好みでもあるのですが、個人的にはスライドの文字は、

どんなに小さくとも20ポイント以上に
したほうが良いでしょう。

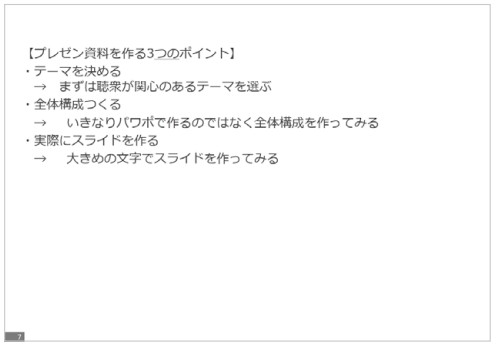
たとえば、悪い例として**図5**を、良い例として**図6**を挙げておきます。



9th Stage 読み上げるだけのスライドに意味はありません!

もう1つスライドの文字数を少なくしたほうが良い理由として「文字だけの資料」はプレゼンを聞いている人に伝わりにくいというものがあります。なぜなら、

▼図5　悪い例



▼図6　良い例

スライドに文章が書いてあると、
それを読み上げてしまう

からです。

確かに、スライドにこれから説明する内容が書いてあれば、発表者はそれを読み上げるだけですので、プレゼンの最中は非常に楽だと思います。ただ、これから発表する内容がすべてスライドに書いているのであれば、プレゼンをする意味はありません。

単にスライドの内容を読み上げるだけであれば、プレゼンをするのではなく、スライドの内容を印刷して各自で読んでもらったほうがはるかに理解してもらえると思います。わざわざプレゼンをするということは、スライド資料を読み上げる以上の説明をする必要があります。より相手に自分の訴えたいことをわかってもらうには、

図表やイラストを多用すると良い

のです(図7)。スライドや講演資料を作る際、とくに時間がないと、どうしても文字だけになりがちです。ただ、文字だけのスライドを作る場合でも「長い文章」ではなく、要点だけを大きな文字で書き(図8)、それを説明できるようにしておきましょう。

うけるプレゼン　視覚的な素材を多用する

すべるプレゼン　スライドを読み上げてしまう

10th Stage
表紙にこだわっていますか?

プレゼンの資料で一番見られるのは、どこだと思いますか?

一番見られるのはプレゼンで利用する
スライドの「表紙」

です。自分のプレゼンが開始される前には、自分のプレゼンの表示が投影されるケースが多いでしょうし、プレゼンを行ったあとで資料を公開する場合プレゼンの「表紙」が一番見られます。

スライドの「表紙」が
一番長く見られる可能性

があるのですからスライド資料の中でも、

「表紙」は、
こだわって作成しましょう。

とは言え、具体的にはどのようにスライドの「表紙」を作ったら良いでしょうか? いろいろな方法がありますが、高画質の写真素材などを利用して表紙を作るのがお勧めです。何しろ、一番長く見られるスライドですので、

文字だけの殺風景な表紙よりも
きれいな写真やインパクトがある画像

を使用したほうが、聴衆の目にとまりやすいでしょう。たとえば筆者の場合を図9に示しました。

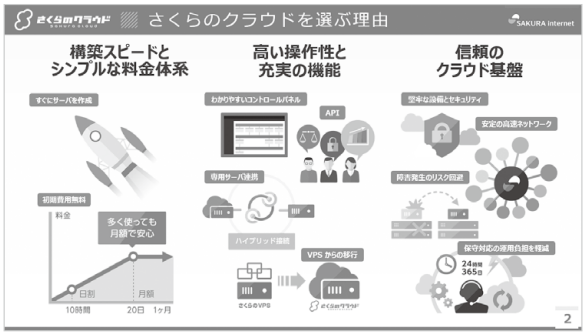
スライドの表紙に利用する写真やイラストの探し方ですが、表紙に利用するものは、できれば有料の写真素材配布サイトから探すのが良いでしょう。最近では、フリーの写真素材やイラスト配布サイトも増えてきましたが、無料のサイトでは目的の画像が探しにくいということもありますし、人気のある無料の写真・イラスト配布サイトの物は多くの人達が利用しているので、どうしても「ありきたり」なスライドになってしまいます。

一方、有料の写真素材は、有料だけあって品質は高いですし、そこまで多くの人達が見ているわけではありません。さすがにすべてのスライドで有料の写真素材を利用することは金額的に難しいですが、

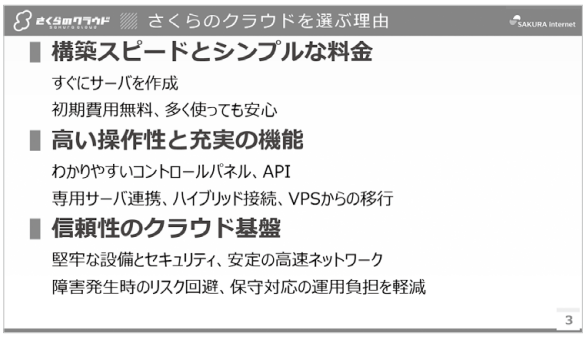
スライドの表紙で利用する素材は
有料素材を選択

したほうが良いでしょう。

▼図7　図表やイラストが多いスライド



▼図8　長い文章ではなく、要点を絞る



うけるプレゼン　スライドの表紙の美しさにこだわる、素材も有料のものを使う

すべるプレゼン　スライドの表紙で手を抜く

11th Stage その「ネタ画像」は本当に必要ですか？

技術系の勉強会では、さまざまな人達が発表します。中には、

アニメやマンガの一場面を
切り出した画像や受け狙いの「ネタ画像」

を入れる方もいるでしょう。自分のプレゼンに関心を持ってもらうという点では、これらの画像を利用することは悪くはありません。

とくに自分のプレゼンを聞いている人達がアニメやマンガに興味がある場合や、勉強会の雰囲気がそれほど「お堅い雰囲気」ではない場合は、いろいろとネタ画像を仕込んで、自分のプレゼンに関心を持ってもらうことは良いことです。

しかし「ネタ画像」を多用して、
プレゼンで笑いを取ることに
注力してしまう

と、肝心のスライドの中身を作り込めなくなってしまいますし、集めたネタ画像をどうしても使いたくなり、

そもそも何のためにプレゼンを
しているのかが、わからなくなって

しまいます。

また、自分がおもしろいと思ったことでも、あなたのプレゼンを聞いている人達が、それをおもしろいと思うかどうかはわかりませんし、プレゼンをするあなたはユーモアのつもりでも逆に怒り出す人もいるかもしれません。

プレゼンで、自分のプレゼンに関心を持ってもらうために適度なユーモアは効果的かもしれませんが、

▼ 図9　筆者のスライド表紙例（石狩データセンターとクラウド）

別にプレゼンをする人は
お笑い芸人ではありません。

ネタ画像を入れることや聴衆を笑わせることは目的ではなくあくまで手段です。これらは使いどころを考えながらプレゼンを行うと良いでしょう。

うけるプレゼン　受け狙いは最小にとどめる

すべるプレゼン　ネタ画像を多用する

column コラム

ネタ画像の弊害

たとえば、プレゼンの当日、Twitterなどで面白い画像を見つけました。なんとなく受けそうなので、使いたくなってしまいます。しかし、プレゼン資料を見直すと、その画像を入れる場所がありません。その場合、無理に利用することはないでしょう。

また、ネタ画像を自分で作るのであればともかく「ネット上から拾ってきた画像」の場合、著作権的にグレーな場合があります。終了したプレゼンをslideshareなどで公開することもあると思いますので、そもそもそういう画像は入れないほうが良いと思います。

なお、どうしても「インフラ系エンジニアの勉強会でよく見るネタ画像」を探したいという方は「汎用性の高い画像」あたりで画像検索するといろいろと出てくると思います。

Lesson 4 成功ノウハウ 編 ―― より良い発表の仕方

1st Stage 会場には早めに着いてプロジェクターの確認をする

プレゼン資料ができて発表の練習をしていたら、とうとうプレゼンの当日になりました。あとは自分の発表をするだけです。ここでは、発表当日にどのようなことを意識してプレゼンをすれば良いのか説明をしていきたいと思います。

まず、プレゼンの会場では、

できるだけ早めに着いて
準備をすること

をお勧めします。これは「遅刻しない」といった基本的なこともありますが、プレゼン当日に、発生する可能性があるさまざまなトラブルに対処するためです。会場に早めに着けば、これらのトラブルに対処できます。プレゼン時のトラブルで、とくに多いのが、

会場のプロジェクターと自分の持ってきた
パソコンとがうまく接続できないトラブル

です。早めに会場に着いて、自分のパソコンとプロジェクターの確認をすることをお勧めします。もし、うまく接続できなければ、主催者の人からパソコンを借りて、そのパソコンでプレゼンをするように調整をする必要があります。そのために、自分が作成したプレゼン資料は自分のパソコン以外にも、

USBメモリやDropboxなどの
オンラインストレージに保存

しておきましょう。プロジェクターとの接続がうまくいっても、念のためにスライドを、

全画面モードにして、
正常に投影されるか確認

しておいたほうが良いです。とくにPowerPointの発表者ツールなど発表者用の画面とプレゼンテーション画面を別々に設定している人の場合は、発表時になぜか、

スクリーンには発表者用の画面、
パソコンにはスライドの全画面と
本来とは逆のスライドが表示

される場合もあります。そうならないように確認をしておきましょう。

このほかにも会場に早く着いたら、

プレゼンを行う会場の広さやマイクの
有無、ネット環境などを確認

しておきましょう。もし、会場が広くマイクがないのであれば、プレゼン中大きな声でしゃべらなければなりませんし、デモをしようと思っていたのにネット環境がなければ、スマートフォンでテザリング接続する必要が出てくるかもしれません。

これらは、細かいことではありますが、このようなトラブルが起こると、あわててしまいトラブル解消に時間がかかってしまいます。あわてないためにも事前に確認をしておきましょう。

うけるプレゼン　会場に早めに着いて準備する

すべるプレゼン　いきなり始めてネットワークやディスプレイ設定で失敗する

2nd Stage 会場にいる参加者を確認する

会場に早めに着いてすることはプロジェクターの確認だけではありません。自分のプレゼンを聞くであろう、

参加している人の様子も確認

しておきましょう。どのような人が自分のプレゼンを聞きに来ているのかがわかれば、プレゼンでどのように話せば良いのか考えることができます。

また、プレゼンのメイン登壇者が自分自身であれば、

会場で目に付いた人と話してみるのも良いかもしれません。

実際に話すことにより、今回のプレゼンで、どのようなことを期待しているかがわかるので、その要望にあった話し方をすれば良いでしょう。そのほかにも、今回のプレゼンの主催者の方に挨拶をして、本日はどのような人が参加しているのか確認をするのも良いでしょう。もし、自分が作っていたスライド資料が会場にいる聴衆の期待と違っていたとしても、スライドを全面的に直すことはできないでしょうが、プレゼンでの話し方は変えられるはずです。

3rd Stage スライドを見て話すな、聴衆を見て話せ

プレゼンをしているときに、あなたが最も見なければいけないものは何でしょうか？　プレ

column コラム

聴衆が聞きたいテーマがわからないのであれば、その場で聞いてみる

会社のセミナールームをコミュニティなどにお貸しすると「宣伝枠」としてLTや講演ができる場合があります。せっかくの好意なので普通に会社紹介や商品紹介をしても良いのですが、できれば聴衆が聞きたいテーマを話したいものです。

でも、どうすれば聴衆に関心があるテーマを出せるのでしょう？　その場で、みなさんの声を聞いて、その場で資料を作って発表をすれば良いのでしょうが、さすがにその場で資料を作って発表というのは難しいです。

そこで、筆者は3〜4つぐらいプレゼンのテーマを出して、会場にいる聴衆に選んでもらいました。たとえば、ある勉強会では「①今年のIT業界ニュースベスト5、②IPv6、③会社紹介」の3つを出して、聞きたいテーマのアンケートをとったこともあります（このときは「今年のIT業界ニュースベスト5」の人気が1番でした）。

このように会場で実際に聞きたいテーマをいくつか用意して、聴衆に選んでいただくと、自分の中で「これが選ばれるだろう」と思っていたテーマを聴衆が選ばないときもあります。たとえば分散SNSである「マストドン」が流行し始めたときに、インフラエンジニア向けにプレゼンをする機会がありました。そのときに筆者は「①マストドンの話、②IPv6の話、③プレゼンのやり方」といったテーマの中から選んでもらうような形にしました。

ちょうどマストドンが流行り始めたことや、前日にマストドンのプレゼンをしたときにも、それなりに人が集まっていたので、てっきりマストドン関連のプレゼンが選ばれると思っていたのですが、この中で選ばれたのは、なんと「IPv6」でした。

昨日のエンジニア向けのイベントではマストドン関連のプレゼンがうけたのに、なぜマストドンではないのか気になりましたが、プレゼンが終わったあとにいろいろと聞いてみると「IPv6の話は最近聞かないので気になった」「仕事でいろいろとソフトウェアを触っているので、プライベートの勉強会に来てまでマストドンのような話を聞きたくない」といったご意見をもらいました。

このように、もし、プレゼンのテーマが自由になっているのであれば、聞きたいプレゼンを聴衆に選んでもらうことも有効です。

ゼンに慣れていない人の場合、投影されている自分のスライドや自分のパソコンを見てしまいがちです。しかし、

あなたが本当に見なければ
いけないのは聴衆

です。あなたのプレゼンを聞いているのは自分が作ってきたスライドではなく聴衆です。せっかく自分の話を聞いてもらえるのですから、聴衆を見て話をしましょう。聴衆を見ながら話をすると、前に座っている人の様子はもちろんのこと、後ろに座っている人達の様子も、はっきりと見えるでしょう。会場の全体に目を配りながら、まずは自分のプレゼンに、

「興味を持って聞いている人」を探しましょう。

もし、そのような人がいればしめたものです。その人を中心に会場のほうを見てプレゼンをしましょう。

反対に、プレゼンを聞いている人の中には、あなたのプレゼンに関心がなくあくびをしている人、もしかしたら寝ている人達もいるかもしれません。こういう人達を見てしまうと、プレゼン中でもくじけてしまうかもしれません。しかし、会場の中にはあなたのプレゼンを聞いている人は必ずいます。プレセン中にくじけそうになった場合は、自分のプレゼンに関心を持ってくれそうな人を探し、

その人を中心に話してみましょう。

うけるプレゼン　スライドを見ないで聴衆を見る

すべるプレゼン　スライドばかりを見てしまう

4th Stage 大きな声で発表する

プレゼンをするすべての会場にマイクがあれば良いのですが、50人程度の収容力がある会場でもマイクがない場合があります。こうしたマイクがない会場の場合、発表者の声が小さく聞き取りにくいプレゼンがあります。どんなに良いプレゼンであっても、

発表者の声が聞こえにくければ、
集中してプレゼンを聞いてもらえません。

難しいかもしれませんが、マイクがなくても50名程度入る会場では、

後ろのほうまで聞こえるようにプレゼン

をしましょう。もし、自分の声が届いているか心配ならば、発表の前に後ろの人達に自分の声が届いているかを確認してみましょう。もし自分の声が届いていないようでしたら、自分の声をもう少し大きくして発表しましょう。

うけるプレゼン　マイクがなくても大丈夫

すべるプレゼン　ぼそぼそしゃべってしまう

5th Stage 時間を過ぎてプレゼンをしても誰も聞いてくれない

勉強会やセミナーでプレゼンを見ていると、自分の持ち時間をオーバーしてプレゼンを続けられる方がいらっしゃいます。プレゼンの発表者にとっては、時間内に伝えきれなかったので、もう少しプレゼンを続けたいという気持ちはわかりますが、

プレゼンは時間内に終わらせたほうが良い

でしょう。

これは、セミナーや勉強会のタイムスケジュールを守るということもありますが、時間を過ぎたプレゼンは誰も関心を持たず、そのプレゼンは誰も聞いていないということがあります。もちろん、プレゼンの内容がとてもおもしろく聴衆からももっと聞きたいと思うようなプレゼンであれば時間を過ぎても良いと思いますが、大半のプレゼンではそうではなく、

聴衆の関心は
「このプレゼンがいつ終わるのか？」

ということしかないでしょう。もし、自分のプレゼンが時間どおりに終わらなさそうになったら、後半のプレゼン資料で飛ばせるものは飛ばしてしまいましょう。別に用意したスライド資料のすべてを詳細に説明することはありません。それよりも時間内にプレゼンを終わらせるようにしたほうが良いです。逆に自分のプレゼンが早く終わってしまった場合は、質問の時間を多めにとったり、説明が足りなかった部分の補足をすれば良いでしょう。

6th Stage プレゼンにデモを加えていますか?

スライドで発表するだけがプレゼンではありません。自分が紹介したいものについて、プレゼン中にデモができるのであれば、

実際にデモをしてみましょう。

デモをすれば、実際に動いているところが見えるので、単にスライドで紹介する以上の印象を与えられ、説得力が増します。ただ、プレゼンの最中にデモを行う場合、

失敗するというリスクもあり

ます。プレゼンの3分前までは正常に動いていたのに、いざ本番で動かすときはなぜか正常に動かなかったり、紹介するサービスに障害が発生したりといったことはざらにあります。

デモをする場合、失敗することを想定して、事前に準備をすることをお勧めします。たとえば、プレゼンの中で、あるソフトウェアをインストールするデモを実施するのであれば、事前に完成したものを用意しておき、デモに時間がかかったり失敗したときは、そちらを紹介するなど、失敗に備えて準備をしておきましょう。

もし、可能であれば事前にデモの手順を録画しておき、プレゼンでデモを紹介するときは、それを流すのも1つの方法です。ただし、プレゼンをする会場で、動画の音声が正常に再生されるか確認が必要です。

また、デモを行うときは、単に画面を操作するのではなく、聴衆に解説をしながらデモを行うことも重要です。自分では普段、操作している画面なので気にしないと思いますが、あなたのプレゼンを見る人にとっては、初めて見る画面で、何の画面かわからないかもしれません。

デモをするときは画面の
解説をしながら実施しましょう。

とくにライブコーディングをする人の場合、コーディングに集中するあまり、画面の解説が手薄になる方も多いように思えます。デモの実施中でもスライドを利用して話しているように、解説をしながら操作をしましょう。

Last Stage 終わりに

ここまで、プレゼンについてのテーマ決めや、スライドの作成方法、実際の発表時の注意点などを説明してきました。もしかしたら、どれも基本的なことですので、

「そんなこと言われなくともわかっている」

というところもあったかもしれません。しかし、実際にプレゼンを実施するときに、ここまで書かれていたことをいきなり実践するのは、なかなか難しいでしょう。

「プレゼン中に聴衆を見る」

column コラム

デモ失敗時のリカバリー方法

ただ、どんなに準備してもデモで失敗することはあります。偉そうに「デモをするときは準備しよう」と書いている筆者もよく失敗していました。

とくにデモをするときにコマンドの打ち間違いをすることはよくあります。通常であれば、すぐにわかることでもプレゼン中には、なかなか気がつかないものです。デモに失敗したあとに失敗した個所がわかったり、場合によっては聴衆に「そこの文字が間違っている」と教えてもらうときもありました。

最近では、このようなミスを減らすため、自分がデモをするときにはあらかじめコマンドを書いておいて、それをコピー&ペーストするようにしています。ただ、この方法でもデモ中にコピーするところを間違えたりする可能性もあるので注意が必要です。

また、デモ中の機材関連のトラブルなど、事前準備では防ぎきれないものもあります。昔はパソコンが不調になってブルースクリーンで落ちたりするケースもあったと思いますが、最近でもプレゼン中にパソコンが固まってしまったり、突然、無線LANが切れるといったこともあると思います。

もしトラブルがあった場合でも、デモのときと同じくトラブル解消に集中してしまうことで発表者が黙ってしまうケースもあると思いますが、このときもできるだけ話をしながらトラブル解消をしましょう。「あれ無線がつながりませんね……ちょっと無線の設定を見てみましょうか、このアイコンをクリックすると無線が接続できますよね？」といったように現在の状態を実況中継するだけでも良いです。トラブル時にこそ、話を続けて、できるだけ自分のプレゼンに聴衆の意識を向けるようにしたほうが良いでしょう。

と言われても、プレゼンに慣れていない人にとっては、頭の中ではわかっていても、どうしてもスライドばかり見てしまうと思いますし、

「練習が大事」

と言われても、スライドを作るだけで精一杯でなかなか練習ができないこともあると思います。このようなことができるようになるには、

何度か人前でプレゼンをして
「場慣れ」をする必要

があるでしょう。まだ、それほどプレゼンに慣れていない人ならば、何度かプレゼンで失敗してしまうかもしれません。ただ、

そのような失敗も「場慣れ」を
するためには必要

です。勉強会のLTぐらいであれば、何度か失敗しても、ただ恥をかくだけです。これらを経験しておけば「次のプレゼン」をするときは確実にうまくなっています。失敗することで向上したプレゼン力は、仕事や自分の人生の中で「絶対に失敗できない」プレゼンをするときに必ず役に立つと思います。

まずは、実際に人前でプレゼンをしてみて「場慣れ」をして、自分のプレゼン力を鍛えてみましょう。SD

うけるプレゼン

1. 聴衆を見る
2. 練習を積む
3. いろいろなシチュエーションで場慣れする
4. 失敗は成功の糧
5. プレゼンは自分の成長の バロメータ

ひみつのLinux通信

作)くつなりょうすけ
@ryosuke927

第42回 新人配属キタコレ

印税で空前絶後の豪華なバカンスを計画している天下無双のニートマンガ家、キャッシュカードの暗証番号も927！すべてをさらけ出した「くつな先生」のマンガが読めるのは本誌だけ。

to be Continued

そろそろ研修上がりの若者が、皆さんの部署に配属された頃でしょうか。初々しいことでしょう。若々しくて勢いがあるでしょう。勢いよすぎてコマンドを確認せずにEnterしちゃってトラブルとか起こしちゃうでしょう。ちょっと前まではみんなあんな感じだったのに、今はスレちゃって……ねぇ。若者にとって、失敗は成長の糧です。ちょっとぐらいの失敗は笑って対応、チョチョイのチョイで直してあげれば「頼れる先輩」アピールできますし、新人も育ちます。デキる・バイブスアゲアゲな先輩になりたいですね。新人だけじゃなく、老害にならないように自分も成長しないとね。初々しい季節、あなたも使わず嫌いせずに新しいツールも積極的に試すのはどうでしょう。え、私ですか？　先月号の特集を読んで、今頃Atomを使い始めました。えへっ。

pixivが「Pawoo」を爆速リリースした理由

「Mastodon」旋風からわかるSNSの未来

構築・運用のノウハウから将来像まで

Author 川田 寛(かわだ ひろし)
道井 俊介(みちい しゅんすけ)
ピクシブ㈱

2017年春、あるSNSが突如、注目を浴びました。その名は「Mastodon(マストドン)」。分権型SNSと呼ばれ、個人や企業が運用する複数のMastodonサーバがつながりあって、1つの大きなSNSを形成しているのが特徴です。Twitterの代替として人気を集めていますが、このMastodon旋風が意味するものはそれだけにとどまりません。Mastodonで使われている技術は、じつは今後のSNSの行く末を占うものなのです。

Mastodonとは何か？

Mastodonは、Twitterに代表されるミニブログ型(microblogging)のFLOSS(Free/Libre and Open Source Software)です。特長的なのは、ミニブログ型SNSを特定の人や企業に独占させるのではなく、さまざまな人々によって運用することを許容していく「連合ソーシャルネットワーク(federated social network)」というアイデアに基づいている点です。Mastodonのソースコードは GitHub上に公開されており、誰でもサーバ上にインストールして運用できます。

Mastodonは、

- 短文の投稿(Mastodonでは「トゥート」という)ができる
- フロー型のタイムラインを持つ
- Twitter用クライアント「TweetDeck」に外観が似ている(**図1、2**)

など、さまざまな点でTwitterの影響を感じられます。実際に日本のメディアではTwitterと比較されることが多く、ピクシブ㈱が2017年4月にリリースし、運用を始めた世界最大級のMastodon「Pawoo」(**図3**)でも、多くのユーザがTwitterではできないことを実現する手段として注目しています。

▼図1　TweetDeckの画面

▼図2　Mastodonの画面

▼図3　Pawoo

ただ実態として、Mastodonで得られる体験は、Twitterのそれとは大きく異なっています。

Mastodonの設計思想 ――「脱中央集権型」という考え方

Twitterは、例えるなら無色です。いろんな国・趣味を持った人たちが、Twitter社だけで運営するTwitterという1つのシステムの中に押し込められている「中央集権型」という設計で運用されています。フォローやキーワード検索、タグを駆使して、自分にとって関心が高いツイートが集まっているタイムラインを眺めるという楽しみ方をします。

一方でMastodonは、インストールされたサーバ（以下、インスタンス）が、2017年6月現在、2,500以上運用されています。運営元には、ピクシブ㈱のような企業もいれば個人もいます。そして、お絵かき好きが集まる「Pawoo」、音楽ファンが集まる「Pawoo Music」、エンジニアが集まる「Qiitadon」など、それぞれのインスタンスが強いカラーを放っています。ユーザは、その中から自分の趣味嗜好（しこう）とマッチしたインスタンスを選んで参加します。

ユーザは、それぞれのインスタンスごとに醸成された強烈なコンテキストを楽しみます。Pawooであれば、絵を描く人の流行や、有名イラストレーターの投稿に全員が一体となって熱狂しています。インスタンスに登録しているすべてのユーザの投稿を一覧することができる「ローカルタイムライン」は、一体感を楽しむためのホームのように活用されています。ホームとして眺めるタイムラインが異なる、それがTwitterとは体験が異なる理由だったりします。

Mastodonのインスタンスは、それぞれが完全に独立しているのではなく、世界中のほかのインスタンスと疎結合でつながり連動しています。別のインスタンスにいるユーザをリモートフォローすれば、そのインスタンスに参加することなくトゥートを眺めることができます。「連合タイムライン」を使えば、インスタンス上のすべてのユーザが、インスタンスを超えてリモートフォローしているすべてのユーザのトゥートを眺めることができます。

つまり、インスタンスのうちどれかに参加すれば、Mastodonという巨大なSNSに参加しているような体験が得られるのです。

それぞれのインスタンスが強烈なコンテキストを持ち独自進化しつつ、お互いがゆるくつながることでMastodonという巨大なSNSを形成していく。それこそが、Mastodonが提唱する「分権型（＝脱中央集権型）」という設計思想であり、それを支えているのが、「連合ソーシャルネットワーク（federated social network）」の技術です。そして、そのインスタンス同士の連携を支えているのは、あのHTML5のムーブメントを起こしたW3C（World Wide Web Consortium）発の標準技術だったりします。

Mastodonの歴史 ――「OStatus」という標準技術

2008年、さまざまなブログシステムが出現する中で、複数の対等なインスタンス間で仕様がオープンなAPIを使って連携する、分権型のSNSが生まれました。それが「Identi.ca」というTwitterクローンです。「OpenMicroBlogging」という仕様を公開し、それに準拠したSNS同士で連携できるようにしました。

2010年頃、OpenMicroBloggingは「OStatus」へと引き継がれます。HTML5が注目を集め始

めた2012年にはW3CにてCommunity Groupが発足され、同仕様の検討が進められます。2014年には、「Social Web Working Group（Social WG）」というSNS向けのさまざまな仕様の標準化を進めるグループが発足されます。そして、その枠組の1つとして、OStatusはリアルタイムにSNS同士の連動を行うための標準として議論されます。

OStatusとは何か？　一言で言うなら、ミニブログ型SNS同士で投稿やプロフィールなどの情報をリアルタイムにやりとりするためのAPIのルールです。次のようなプロトコルやフォーマットなど、さまざまな仕様の集まりです。

- ユーザの投稿を別のインスタンスへ通知する「PubSubHubbub」
- 投稿に対する返信やリアクションを、インスタンスを超えて行うための「Salmon」
- インスタンスを超えてアカウント情報のやりとりを行う「WebFinger」
- 個々のコンテンツを表現するフォーマット「Portable Contacts（POCO）」「Atom Activity Streams」「Atom Threading」

このように、OStatusはさまざまな仕様を組み合わせることで、独自実装に頼らずにインスタンス同士を連携させることができます。

ただ、OStatusができた当初、なかなか世の中に受け入れられませんでした。そうして議論は進まなくなり、もはや死んだに等しい状況に陥ります。同じ時期に、GoogleもOpenSocialというほぼ同じアイデアを考えていましたが、こちらもうまくいかず消滅してしまいます。

一方で製品側では、Identi.caは「StatusNet」へと名を変え、インスタンス間の連携方法もOpen MicroBloggingからOStatusへと移り変わります。同時に、StatusNetはさまざまな実装へとフォークされました。その中で「Free social」と呼ばれる実装も出現します。StatusNetがなかなか世の中に認められず伸び悩む中、StatusNetとFree socialは「GNU Social」へと統合されます。

このOStatusを中心に広がるミニブログ型SNS連携技術に目をつけたのが、ドイツの若きエンジニア、オイゲン・ロッコ（Eugen Rochko）氏です。彼は、PHP実装だったGNU SocialをRuby on Railsに作り変え、自分の思想に沿った実装に落とし込みました。そして、2016年10月に公開されたのが「Mastodon」です。

7年の時を経て、ようやく日の目を見た連合ソーシャルネットワークの標準技術。「Mastodonの本質はOStatusだ！」と語られることがありますが、その理由はもうおわかりのはずです。

Mastodonとは、いわばOStatusによってつながった巨大なネットワークのクライアントのようなもの。たとえば、OStatusがSMTPだとしたら、MastodonはSendmailやPostfix、qmailみたいなものです。別にMastodonを使うことにこだわる必要はありません。OStatusに準拠したエンドポイントを持つSNSを公開すれば、あなたも簡単にMastodonという巨大なSNSの一部になれます！

Mastodonのコミュニティ——「AGPL」というライセンス

「OStatus準拠のSNS実装があればいい」とは言いましたが、残念なことにOStatusだけでは機能が不足しており、Mastodonでは独自に機能を足しているという実態があります。そのため、生きた実装であるMastodonを使ってインスタンスを立ち上げるのが最速です。

それに人々は、自前のSNSにOStatusのAPIを実装することよりも、Mastodonのソースコードを編集して、デザインを変えたり、独自機能を追加したりなど、インスタンスの特性にあったカスタマイズをして運用していくことに熱狂しています。

そのことにいったいどれだけの自由が許されているのか？　2017年4月、筆者（川田）はGitHubのIssue[注1]を通じて1つの問いを投げかけま

注1）URL https://github.com/tootsuite/mastodon/issues/2007

した。「Mastodonのmasterでは『トラッキングしない』という思想が含まれているが、これはForkしたほかのインスタンスにおいても同様のことを強制しているのか？」。これに対して開発者のオイゲン氏は、「個々のインスタンスでトラッキングなどを導入することは自由だ」と回答しています。

Mastodonのインスタンスは、それぞれが独自の思想を持って進化することを、オイゲン氏自身も許しています。Mastodonはインスタンスの数だけソースコードをForkしていくことが前提のシステムであるとも言えます。こうした点で、1つのソースコードリポジトリをみんなで支えていくという一般的なFLOSS開発とは異なる開発コミュニティが形成されています。

Forkしたことで、Fork元の進化が止まっては元も子もありません。この問題をうまく解決しているのが、Mastodonのライセンスである「Affero General Public License（AGPL）」です。

AGPLでは、サーバ上で運用し、ユーザからアクセスされるようなソフトウェアのソースコードは、開示する義務が生じます。すべてのインスタンス運営者は、GitHubなどのしくみを使って、何かしらの形でソースコードを公開しなくてはいけません。すべてのインスタンスのソースコードに対する変更が、Mastodon全体の資産として扱えるようになっています。

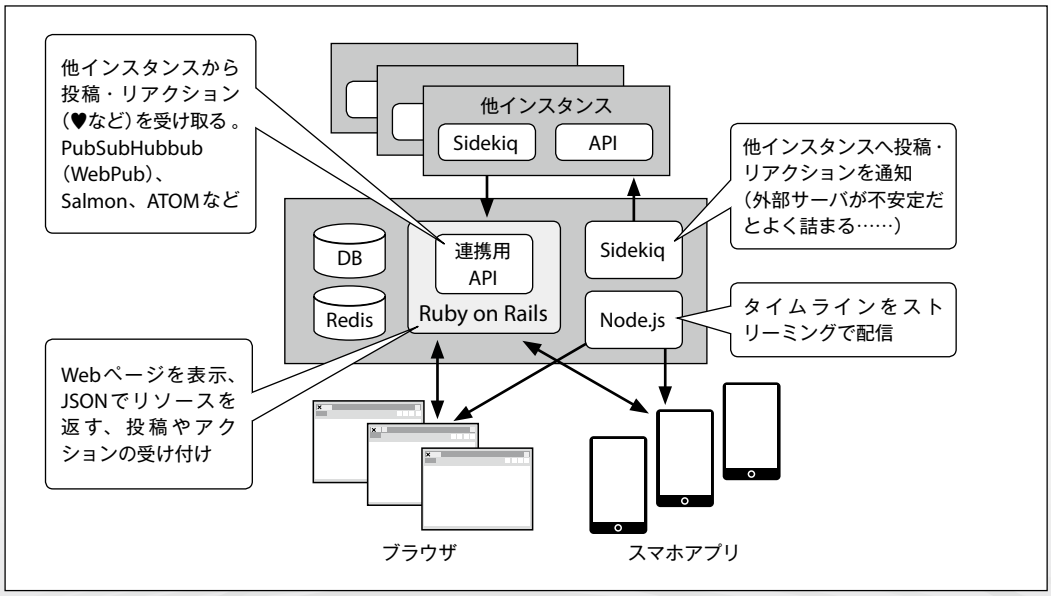
「どうせソースコードは公開しなくちゃいけない。それなら！」と、各インスタンスの運営者も、ただ自分のインスタンスのみを改善するのではなく、有益なものは積極的にオイゲン氏のMastodonリポジトリへPull Requestを投げるような空気が生じています。また逆に、インスタンス側に対してアップデートのPull Requestを投げるような人も現れています。

もしあなたがMastodonのインスタンス運用を始めたなら、そこで見つかった課題や解決案を、どんどんオイゲン氏のMastodonリポジトリへフィードバックしていくといいでしょう！　運営する楽しみのお礼に、Pull Requestで恩返しすることも、Mastodonの醍醐味の1つです！

Mastodonインスタンスの構築

それではMastodonインスタンスを実際に構築する方法について見ていきましょう。**図4**に示

▼図4　Mastodonの全体像



したとおり、MastodonはRuby on Railsを中心としたWebアプリケーションです。

Mastodonは、Webアプリケーションとしてはそれほど複雑な構造のものではありませんが、それでも複数のサーバプロセスを管理し、データベースとKVS(Key-Value Store)を運用しないといけないため、それなりに高度な知識が要求されます。少なくとも一般的なHTTPサーバの運用経験と、Gitを使ったソースコード管理の経験はあったほうがいいでしょう。Ruby on Railsで記述されたアプリケーションの運用経験があれば、より簡単に導入できるはずです。

本稿では、最も簡単にMastodonを運用する方法としてDockerを用いた方法を紹介します。

ソースコード

MastodonはGitHubリポジトリで開発、公開されており、最新の安定版はGitHubリポジトリのリリースページからダウンロードすることができます。執筆時点(2017年6月現在)の安定版はv1.4.3です。

・Mastodonリリースページ
https://github.com/tootsuite/mastodon/releases

後述するDockerイメージは公開されていますが、現時点でパッケージによる公開は行われていません。また、Mastodon自体非常に速い速度で開発が進んでいる状況です。ソースコードの変更に追従しやすくするには、リリースページからZIPファイルをダウンロードするのではなく、リリースタグを指定してGitリポジトリをチェックアウトするほうがいいでしょう。たとえば、v1.4.3ブランチは次のようにしてチェックアウトできます。

```
$ git clone git@github.com:tootsuite/mastodon.git
$ cd mastodon
$ git checkout v1.4.3 -b v1_4_3
```

Gitリポジトリを手元にクローンしておけば、`git fetch -t`を実行することで新しいリリースタグを取得し、ソースコードの変更点を確認することができます。また、自分でインスタンスをカスタマイズする際にも役に立つでしょう。

MastodonはAGPLでライセンスされており、すべての変更に対して公開する義務がありますが、Gitリポジトリであれば簡単にGitHubでForkして公開することが可能です。AGPLであってもMastodonの利用が進んだ要因の1つには、GitHubのForkと相性が良いことがあるかもしれません。

システム構成

さて、ソースコードを眺めてすべて理解できるのであれば必要ありませんが、Mastodonを運用するためには、最低限どのようなプロセスが、どういった役割をもって動いているかを理解する必要があります。v1.4現在のMastodonは**表1**のサーバプロセスから構成されています。

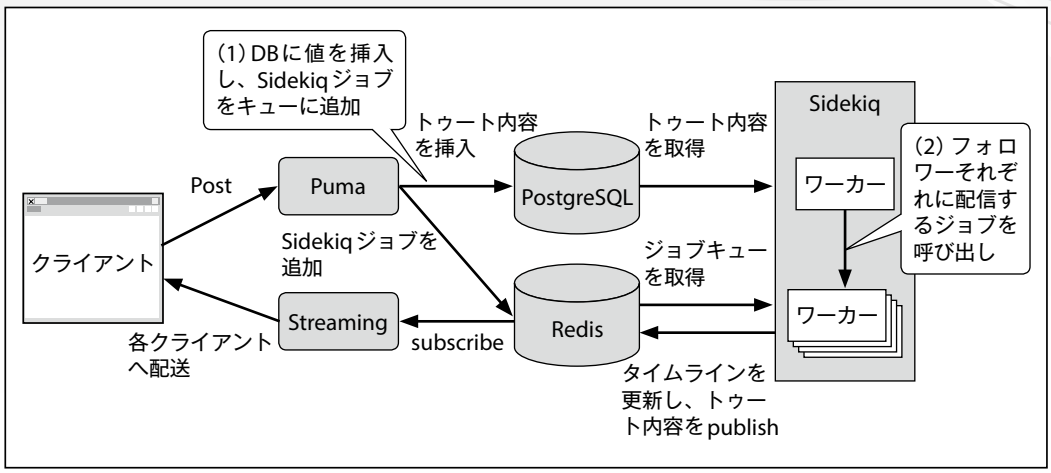
Mastodonではトゥートのリアルタイム配信のために非同期処理を多用しています。Web画面からトゥートされたときの動きを見てみましょ

▼表1　Mastodonを構成するサーバプロセス

プロセス	説明
Puma	Ruby on Railsアプリケーションを動作させるRack[注2]サーバ
Sidekiq	Ruby on Railsでよく利用される非同期ジョブキューサーバ
Node.js	サーバサイドJavaScriptエンジン。MastodonではWebSocketサーバを動作させるために利用されている
Redis	ハッシュのほかにリストやソート済みセットなどさまざまなデータ型を利用できるKVS
PostgreSQL	Ruby on Railsと相性が良いRDBMS

注2) Rubyで一般的なWebサーバインターフェース。Ruby on RailsもRackを利用しています。

▼図5　トゥートが追加された際の処理（ローカルインスタンスの処理のみ）



う（**図5**）。

まず、PumaのAPIエンドポイントにトゥート内容がPOSTされます。するとDBに内容が記録されると同時に、Sidekiqの非同期ジョブが実行されます。呼び出されたSidekiqジョブは、フォロワーそれぞれに配信するジョブを実行します。そして、それぞれのフォロワーに対応するジョブがRedis上のタイムラインの配列にトゥートのIDを追加し、さらにストリーミング配信用のJSONを生成するジョブを実行します。最後にストリーミング配信用のJSONが生成され、Redisに追加され、それをストリーミングAPIを担当するNode.jsサーバが各クライアントに配送することで、ようやくトゥートが通知されるわけです。

Dockerを用いた構築

このように、Mastodonはリアルタイム更新を実現するために複数のプロセスを利用しています。そのため、少し複雑ではありますが、限られた人数で運用するのであれば、これらを1つのサーバで動かしてしまえば問題ありません。幸いなことに、Mastodonには簡単に動作させるためのしくみがいくつか用意されています。その中でも、Dockerを使った経験があれば、Dockerを使うのが最も簡単な方法でしょう。

★docker-compose.yml

まずは、チェックアウトしたMastodonのGitリポジトリにあるdocker-compose.ymlを開いてみましょう。**表1**で紹介した5つのサーバプロセスがそれぞれ別のサービスとして定義されているのがわかります。

ここで最も重要なのは各サービスのボリュームの設定です。ボリュームはDockerコンテナの外の領域をマウントするための設定で、データの永続化に利用されています。Mastodonでは標準で、dbサービスとredisサービスのボリュームがコメントアウトされ無効化されていますが、これを有効にしないとDockerコンテナを再起動した際にすべてのDBの内容が失われてしまうことになります。必ずこの設定を確認するようにしましょう。

また、Dockerのボリュームはコンテナ外部の領域をマウントするため、あまりパフォーマンスが良い方法ではありません。DBやRedisはとくにI/Oのパフォーマンスが要求される部分です。docker-compose.ymlからdbとredisサービス自体をコメントアウトしてしまい、これらはDockerコンテナの外で動かすことも検討しましょう。こうすることで、操作を誤って重要なデータを消してしまうことも防げます。

★.env.production

Mastodonのサーバ設定は.env.productionファイルに記述します。.envは環境変数を羅列するファイルで、.env.productionは本番環境(production)用の設定を記述するファイルです。サンプルとして.env.production.exampleファイルがリポジトリに含まれているので、これをコピーして作成しましょう。おもに設定する項目には次の項目があります。

①DB、Redisサーバの設定
②ホスト名の設定
③アプリケーションシークレットの設定
④メールサーバの設定

「①DB、Redisサーバの設定」では、接続先のRedis、DBサーバのホスト名、接続設定を行います(**リスト1**)。サンプルファイルはそのままDockerの構成に利用できるように設定されています。Redis、DBをDockerの外で動かす場合には、この値を環境に合わせて設定しましょう。その際、DBには専用のデータベースを作成しておいたほうがいいでしょう。

「②ホスト名の設定」では、LOCAL_DOMAINにMastodonインスタンスのホスト名を指定します(**リスト2**)。Mastodonは各インスタンスのトゥートやアカウントをホスト名で区別します。そのため、あとからLOCAL_DOMAINの設定を変更するとさまざまな問題の原因になり得ます。注意して設定し、変更しないようにしましょう。

次に「③アプリケーションシークレットの設定」です(**リスト3**)。これらの値はアプリケーション内で秘密のトークンとして使用されるものです。これらのトークンは次のコマンドを実行することで生成できます。

```
# docker-compose run --rm web rake secret
```

最後に「④メールサーバの設定」ですが、Mastodonは新規登録やパスワードリマインダ、そして通知のためにメールを利用します。そのため、別途SMTPで利用できるメールサーバを用意する必要があります(**リスト4**)。簡単に用意できるサービスにはmailgun、SendGrid、そしてAmazon SESなどがあるでしょう。もちろん、自分でSMTPサーバを立てて運用しても問題ありません。

★Mastodonの起動

さて、ここまで設定が終わればMastodonを起動しましょう。docker-compose up -dコマンドを使って起動します。

▼リスト1　DB、Redisサーバの設定

```
Redisに関する設定
REDIS_HOST=redis          ←Redisサーバのホスト名
REDIS_PORT=6379           ←ポート番号

DBに関する設定
DB_HOST=db                ←PostgreSQLサーバのホスト名
DB_USER=postgres          ←ユーザ名
DB_NAME=postgres          ←データベース名
DB_PASS=                  ←パスワード
DB_PORT=5432              ←ポート番号
```

▼リスト2　ホスト名の設定

```
LOCAL_DOMAIN=example.com  ←インスタンスで使用するホスト名
LOCAL_HTTPS=true          ←HTTPSを利用するかどうか
```

▼リスト3　アプリケーションシークレットの設定

```
PAPERCLIP_SECRET=         ←アップロードファイルのURLシークレット
SECRET_KEY_BASE=          ←セッションの秘密鍵のベース
OTP_SECRET=               ←多要素認証に使う秘密鍵
```

▼リスト4　メールサーバの設定

```
SMTP_SERVER=smtp.mailgun.org              ←SMTPサーバのホスト名
SMTP_PORT=587                             ←ポート番号
SMTP_LOGIN=                               ←ユーザ名
SMTP_PASSWORD=                            ←パスワード
SMTP_FROM_ADDRESS=noreply@example.com     ←FROMに使用するメールアドレス
```

```
# docker-compose up -d
```

★DBマイグレーションとアセットファイルの作成

Mastodonが起動したらDBのマイグレーションとアセットファイルのビルドを行う必要があります。

```
DBのマイグレーション
# docker-compose run --rm web bin/rails ➥
db:migrate

アセットファイルのコンパイル
# docker-compose run --rm web bin/rails ➥
assets:precompile
```

nginxによるプロキシ設定

デフォルト設定の状態では、MastodonはHTTPS通信や静的ファイル配信を行わない設定になっています。これらの処理はHTTPプロキシで行うのが一般的です。

誌面の都合ですべての設定を記述することはできませんが、MastodonのProduction Guide[注3]にnginxの設定例が記述されています。これに従って設定するといいでしょう。

SSL証明書はLet's Encrypt[注4]を使うことで無料で取得できます。Production Guideのサンプルは Let's Encryptを使うことを前提に記述されているため、それに従えば簡単にHTTPS通信を提供できます。

Pawooにおける Mastodonの最適化、改善

弊社が運用している「Pawoo」は世界最大級のMastodonインスタンスの1つであり、1台のサーバでさばけるリクエスト数ではありません。当たり前ですが、複数台のサーバで運用することになります。

クラウドサービスにおける運用

Pawooの構成を**図6**に示します。Pawooではデータベースや Redis、ロードバランサにAWSのマネージドサービスを用いています。AWSではマルチAZ構成をとることでサービスを冗長化し、可用性を確保するのが一般的な構成です。RDSや ElastiCache、Application Load Balancer（ALB）といったマネージドサービスを用いることで、簡単にマルチAZ構成を組むことが可能です。

前述したようにMastodonでは、設定ファイルの接続先DBや Redisサーバの項目にRDSや ElastiCacheを指定するだけで、簡単にAWSのマネージドサービスを利用できます。

Streaming処理の改善

Pawooではインフラサイドの改善以外にも、アプリケーションの改善も行うことでMastodonの処理速度を向上させています。最も効果があったものの1つである Streamingサーバのマルチプロセス化を紹介しましょう。

MastodonのStreamingサーバは Node.jsで実装されています。Node.jsはシングルスレッドで動作するイベント駆動型のサーバプロセスです。そのため常に1コアしか使用できず、ボトルネックになっていました。ここを改善しない限り、Mastodonをこれ以上スケールさせられません。

これを解決する方法として Node.jsプロセスをクラスタリングしてマルチコアを利用できるように改善しました。Node.jsにはプロセスをクラスタリングするための clusterモジュールが標準でバンドルされています。

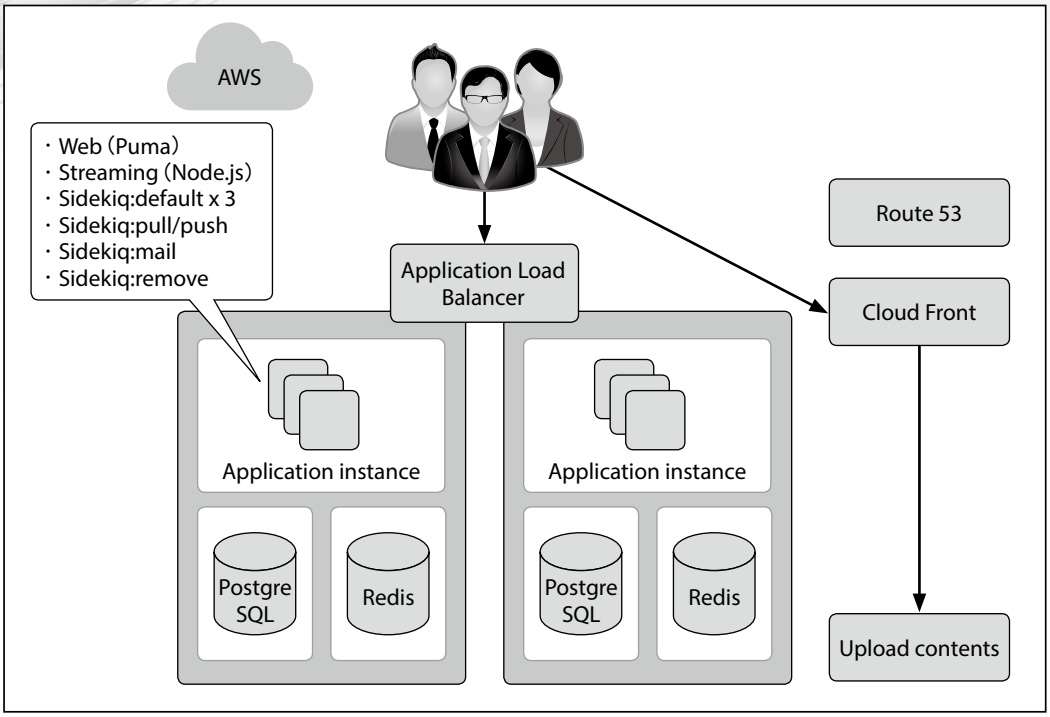
この改善はすでにMastodon本体にも取り込まれており、みなさんは何もしなくてもこの改善の恩恵を受けられます。これ以外にもPawooで行った改善の多数はMastodon本体にすでに組み込まれており、みなさんは意識せずに利用できます。

Mastodonは分散システムであり、Mastodon

注3）URL https://github.com/tootsuite/documentation/blob/master/Running-Mastodon/Production-guide.md

注4）すべてのWebサーバへの接続を暗号化することを目指したプロジェクトおよび認証局。TLSのX.509証明書の発行を無料で行っている。

▼図6　Pawooの構成



がスケールするためには1つのインスタンスだけが改善されてもしかたがありません。Pawooの改善をMastodon本体にもポーティング（移植）していくことで、Mastodonの構築・運用がより簡単にできるようになり、Mastodonコミュニティがさらに発展することを願っています。

なぜMastodonは注目を集めているのか？

GitHubを遡（さかのぼ）ると、オイゲン氏がパブリックな場で実際の作業を始めたのが2016年2月14日、ファーストリリースは同年の10月です。そして今年の3月末から一気に注目を集めます。

4月に入ると、国内では当時筑波大学の学生だったぬるかる氏が個人的に構築したインスタンス「mstdn.jp」が注目を集め、企業が運用するインスタンスとしては初となるピクシブ㈱の「Pawoo」が注目されます。

そのどれもが、Twitterというサービスの影響によるものです。Twitterとは楽しみ方こそ違うのですが、人々がTwitterよりもより良い場所を追い求めた結果として、Mastodonが注目を集めてきたのは事実です。

ツールとしてのMastodonは、間違いなくTwitterにはできないことを実現できる力があり、それによって多くの人が幸せになったのは間違いないでしょう。Pawooにおいても、「グローバル基準で表現の規制がされているTwitterでは、イラストを自由に投稿できない！」そんな悩みを抱えたイラストレーターが自由を求めて集まったのがきっかけです。

ただ、Mastodonのそういう性質だけに注目し続けることは、本質的ではないとも考えています。本当に注目すべきなのは、「Twitterの代替製品が生じたこと」などではなく、「ミニブログ型SNSのオープン化が世の中から受け入れられ始めたこと」だと筆者（川田）は考えています。

Mastodonという箱だけに注目すべきなのか？

あらゆる独自実装がオープンになっていくこ

とは、Free Softwareの登場以来、歴史の中で何度も起きていました。Mastodonの本質的な価値は、ミニブログ型SNSが独自実装と独占から解放され、オープンになる最初のトリガーになったというその1点だととらえています。

ミニブログ型SNSが限られた企業にのみ独占されるのではなく、まるでメールシステムやWebのように、幅広いニーズに応えられる分権化された時代に移り変わろうとしている。それこそが本質的な価値であり、注目されるべきことだと筆者は考えています。

技術がオープン化され、オープンなエコシステムが生み出されようとしている。そういう意味では、議論をMastodonというツールではなく、OStatusなどのオープンな標準とそれによって運用されたネットワークに向けるべきです。

Mastodon/OStatusのこれから

「Mastodonは巨大なOStatusネットワークのクライアントのようなものだ」「Mastodonでなくたっていい」とは語りましたが、実際のところはそう簡単にはいきません。MastodonのOStatusエンドポイントは独自実装ありきで動いており、Mastodonを使って接続するからこそ相互運用できています。

記事の前半で、OStatusは死んだに等しい状況だ、と語りました。今もなおOStatusそのものに大きな変化はないのですが、それを包含したSocial Web Working Groupではちゃんと進化を続けています。Mastodonに関連する代表的な技術について、これからの進化の流れを紹介します。

★ActivityPub

実はOStatus自体を進化させようというモチベーションはすでに失われており、最近は「ActivityPub」がOStatusの後継となることが期待されています。OStatusは良い意味では枯れている技術、悪い見方をすれば化石のような技術をツギハギのように組み合わせているオープン標準の集合体です。一方でActivityPubは、pump.ioやMastodonでも使われているActivity Streamsと呼ばれる仕様をベースに拡張させ、OStatusよりも一貫性があるスペックに仕上がっています。

Mastodonでも、一部だけでも採用できないかと議論されているようですが、あまり前進していません。当面は、OStatusのままでなんとか凌(しの)ごうとしているようにも見えます。

★WebSub

先ほども紹介したPubsubhubbubが元になっており、ここ最近、いろいろと省略されて「WebSub」という読みやすい名前になりました。WebSubはほかのサーバに対して、コンテンツの変更を通知するために使われます。たとえば、ブログの記事を更新した際、それをGoogleの検索エンジンに通知するといった、そういう活用法が期待されているスペック(仕様)です。

Mastodon的な使い方とは、やや異なる需要に応えるべく進化を続けています。Mastodonでは、ほかのインスタンスへ自インスタンスのトゥート内容を通知するために使われています。

★Salmon

WebSubを使って他インスタンスに配信された記事に対して、リプライしたり、★を付けたり、リモートのアカウントをフォローしたりなど、元情報を持つインスタンスに対する何かしらのインタラクションを起こした際に、それを反映させるのに用いるスペックです。情報を持っているインスタンスにどういうインタラクションを起こしたいのかを記述したXMLを、HTTPベースでPOSTするだけというシンプルな仕様です。

「Salmon」はここ最近これといった進化がなく、改善も期待できません。ただ、Mastodonの多くの独自仕様を吸収しており、今後は別のアプローチで改善される可能性を感じます。

★WebFinger

ヨーロッパで多く利用されているOpenID Connectという認証技術を支えるスペックの1つです。Mastodonのような連合ソーシャルネットワークの分野では、他インスタンスからアカウントにひもづいた情報を探すために活用されています。アカウントの命名規則もOpenIDの影響を受けているように見えます。

OpenID Connectのような広く利用されている技術で活用されているというのもあり、まだまだ進化が進んでおり、Salmonと違ってJSONでやりとりできる点でも新しさが感じられます。

★Activity Streams

ブログのFeedでもよく利用されているAtomを改良して作られたスペックで、TwitterやFacebookのような時系列のタイムラインを表現することに特化したフォーマットです。SNSが乱立し、似たようなタイムライン表現が存在する中で、それらを共通のフォーマットで扱えるようにしようとIBMから発案されました。

Mastodonでも他インスタンスからタイムラインを取得するために活用されています。このスペックが、先ほど紹介したActivityPubへと進化しようとしています。

Mastodonだけで進化し続けることの限界

オイゲン氏が運用しているMastodonのmasterブランチには、セキュリティ改善や機能強化など、さまざまな変更が日々繰り返されているため、そこに追従していくことが求められています。差分が大きくなるほどアップデートが困難になるため、実際の運用では乖離（かいり）し過ぎないようにする努力が必要です。

こうした技術的な理由もあり、Mastodonインスタンスを完全には好き勝手にカスタマイズできないという悩みがあります。乖離し過ぎたインスタンスの中には、アップデートされないまま運用されることもあるようです。

それに、なんだかんだ言っても、MastodonはMastodonであり、Mastodonらしさの域をぬけることは困難です。連合ソーシャルネットワークは多くの可能性を秘めていますが、Mastodonであるがゆえに、限られた可能性の中にとらわれてしまうという悩みは一生ついてまわるでしょう。

Mastodonだけを進化の道ととらえない

既存のSNSが段階的にOStatusなどのオープンな標準に対応していくことで、Mastodonを中心とした連合ソーシャルネットワークが巨大化するという道もあり得るでしょう。むしろ、その方向が明るい未来に進めると筆者は考えています。Mastodonという枠ではなく、エコシステムを育てるべき、という考え方です。

たとえば、ピクシブ㈱の例を挙げます。当社が運営しているサービスでは、イラストコミュニケーションサービス「pixiv」という、いわゆるフォトログ型と呼ばれるSNSが有名です。その一方で、時代の移り変わりに応じて、絵を描く人のニーズの変化に応えるべく「pixiv Sketch」と呼ばれるSNSも提供しています。

ミニブログ型であることを追求してきた同サービスは、OStatusやActivityPubに対応することがほぼ可能であり、将来的にそうなることも視野に入れています。もし対応すれば、pixiv Sketchへ投稿したイラストがMastodonインスタンス上でも楽しめたり、また逆にMastodonに投稿されているpixiv文化にマッチしたイラストをpixiv Sketch上でも楽しめたり、といったことができるでしょう。

分権型SNSや連合ソーシャルネットワークというアイデアでブレークスルーを起こしていくなら、Mastodonだけではなく、多くの実装が生まれることが求められるでしょう。それこそ、連合ソーシャルネットワークを取り巻くオープン標準が、かつてのHTML5で起きたような巨大なムーブメントを起こしていくうえで、必然であると筆者は考えています。SD

バックナンバーのお知らせ　BACKNUMBER

バックナンバー好評発売中！常備取り扱い店もしくはWebより購入いただけます

本誌バックナンバー（紙版）はお近くの書店に注文されるか、下記のバックナンバー常備取り扱い店にてお求めいただけます。また、「Fujisan.co.jp」（http://www.fujisan.co.jp/sd）や、e-hon（http://www.e-hon.ne.jp）にて、Webから注文し、ご自宅やお近くの書店へお届けするサービスもございます。

2017年7月号

第1特集
もっとbashを使いこなしませんか?
理論&応用でシェル力の幅を広げる

第2特集
データの抽出・加工に強くなる!
MySQL［SELECT文］集中講座

一般記事
・ハッシュ関数を使いこなしていますか?（後編）
・Jamesのセキュリティレッスン【10】
・Windows Server 2016で構築する最新ファイサーバ（後編）

定価（本体1,220円+税）

2017年6月号

第1特集
Vim、Emacs、Atom、Visual Studio Code
あなたのプログラミングを加速させるエディタ

第2特集
多用途に使いこなせ、コードが読みやすく保守しやすい
今すぐはじめるPython

一般記事
・ハッシュ関数を使いこなしていますか?（前編）
・Windows Server 2016で構築する最新ファイサーバ（前編）

定価（本体1,220円+税）

2017年5月号

第1特集
先輩が教える㊕ノウハウ
Linux入門【UNIXネットワーク編】

第2特集
サイボウズ流　サービス改善につなげる
ドッグフーディング環境の作り方

第3特集
いまから学ぶブロックチェーンのしくみ

定価（本体1,220円+税）

2017年4月号

第1特集
基礎から学べば役に立つ!
目的から考える! 実作業から学ぶLinux入門（OS操作編）

第2特集
AWS LambdaとMicrosoft Azure Functionsで体験
はじめてのサーバーレスアーキテクチャ

一般記事
・mBaaSのしくみ紹介

定価（本体1,220円+税）

2017年3月号

第1特集
マーケティング&サービス向上に役立つ
ログ＆データ分析基盤入門

第2特集
CIでインフラ開発を効率化
Jenkins＋Gerrit＋Ansible＋Serverspecで基盤構築

第3特集
どうなってる? なりすましメール対策
DKIMとホワイトリストによる安心の可視化

定価（本体1,220円+税）

2017年2月号

第1特集
なぜプログラマの役に立つのか
いまはじめるDocker

第2特集
Linuxファイルシステムの教科書
Ext3、Ext4、XFS、F2FS、Btrfsの特徴と進化

第3特集
エンジニアが採用できない会社と評価されないエンジニア（後編）

定価（本体1,220円+税）

Software Design　バックナンバー常備取り扱い店							
北海道	札幌市中央区	MARUZEN＆ジュンク堂書店　札幌店	011-223-1911	神奈川県	川崎市高津区	文教堂書店　溝の口本店	044-812-0063
	札幌市中央区	紀伊國屋書店　札幌本店	011-231-2131	静岡県	静岡市葵区	戸田書店　静岡本店	054-205-6111
東京都	豊島区	ジュンク堂書店　池袋本店	03-5956-6111	愛知県	名古屋市中区	三洋堂書店　上前津店	052-251-8334
	新宿区	紀伊國屋書店　新宿本店	03-3354-0131	大阪府	大阪市北区	ジュンク堂書店　大阪本店	06-4799-1090
	千代田区	書泉ブックタワー	03-5296-0051	兵庫県	神戸市中央区	ジュンク堂書店　三宮店	078-392-1001
	千代田区	丸善　丸の内本店	03-5288-8881	広島県	広島市南区	ジュンク堂書店　広島駅前店	082-568-3000
	中央区	八重洲ブックセンター本店	03-3281-1811		広島市中区	丸善　広島店	082-504-6210
	渋谷区	MARUZEN＆ジュンク堂書店　渋谷店	03-5456-2111	福岡県	福岡市中央区	ジュンク堂書店　福岡店	092-738-3322

※店舗によってバックナンバー取り扱い期間などが異なります。在庫などは各書店にご確認ください。

デジタル版のお知らせ　DIGITAL

デジタル版Software Design 好評発売中！紙版で在庫切れのバックナンバーも購入できます

本誌デジタル版は「Fujisan.co.jp」（http://www.fujisan.co.jp/sd）、「雑誌オンライン.com」（http://www.zasshi-online.com/）、「Gihyo Digital Publishing」（https://gihyo.jp/dp）で購入できます。最新号、バックナンバーとも1冊のみの購入はもちろん、定期購読も対応。1年間定期購読なら5%強の割引になります。デジタル版はPCのほかにiPad／iPhoneにも対応し、購入するとどちらでも追加料金を払うことなく、読むことができます。

人工知能時代のLispのススメ

～ラムダ式からLispの作り方まで

第2回 Lispは人工知能と関数型の先祖！オブジェクト指向、再帰、ラムダ式を学ぼう！

Author 五味 弘（ごみ ひろし） 沖電気工業㈱

Lispは、ラムダ計算を基礎とする関数型プログラミングの先祖です。そして初期の人工知能プログラムの多くはLispで開発されていました。この意味では、Lispは人工知能の先祖でもあります。またオブジェクト指向の発展に寄与し、強力なマクロがあります。今回はLispが語りたいことを紹介します。

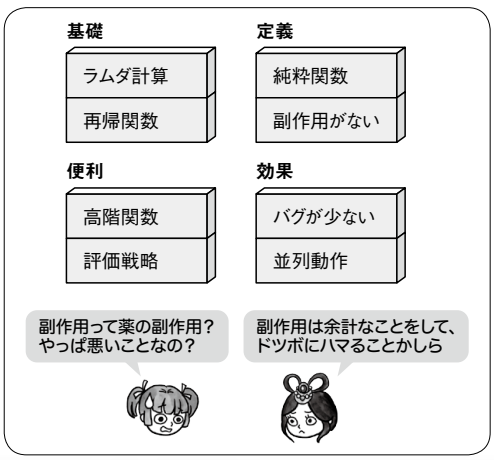
関数型プログラミングの世界の先祖

Lispは世界初の関数型プログラミング言語で、関数型の基本となるラムダ計算と再帰関数（後述）を初めて実装し、後世に続く関数型言語のML[注1]やHaskell、F#、Scalaなどの先達となり、関数型の世界を築き上げました。

関数型プログラミングは、数学の関数（プログラミング言語の偽物の関数ではなく、数学の純粋関数）の考え方をそのまま実装したものです。「純粋関数」とは、いつでもどこでも誰が計算しても、同じ入力に対しては同じ結果が返るものです。このいつでも同じ結果が返ることが重要で、この特徴から、実行順序に影響されることもなく、ほかのプログラムの動作にも影響されることもありません。このようにほかから影響されないため、並列処理でもプログラム置換でも何でもお気楽に自由気ままに実行できます。純粋関数として実行できるのは「副作用[注2]」がないからです。関数型プログラミングでは、この副作用に起因するバグがありません。たとえば副作用によって、並列処理のロック変数が不用意にセットされてしまい、お互いが待ち状態になる「デッドロック」や、逆にロックし忘れで処理が衝突するようなバグもなくなります。IoTやクラウド、ビッグデータなどで並列処理とデータの位置透過性（プログラムやデータをどこに配置しても動作する性質）が求められる時代に、この関数型プログラミングは必須です。この関数型プログラミングの世界を切り開いたことがLispの自慢です（**図1**）。

副作用がないプログラムは、変数に対する破壊的再代入（初期化ではなく、値が格納されている変数に対する代入のこと）をしないことで作れます。**リスト1**に副作用のない関数と副作用のある関数を紹介します。

▼図1 関数型プログラミングはLispが元祖!



注1） Meta-Language。1970年代にエディンバラ大学のロビンミルナーらが設計した関数型言語。

注2） Side Effect。第1回（2017年5月号）の記事でも紹介したが、副作用とは本来の関数の作用でないもので、たとえば変数の値などの状態変化をさせる代入文が副作用の原因になる。副作用は多くのバグの温床にもなっている。

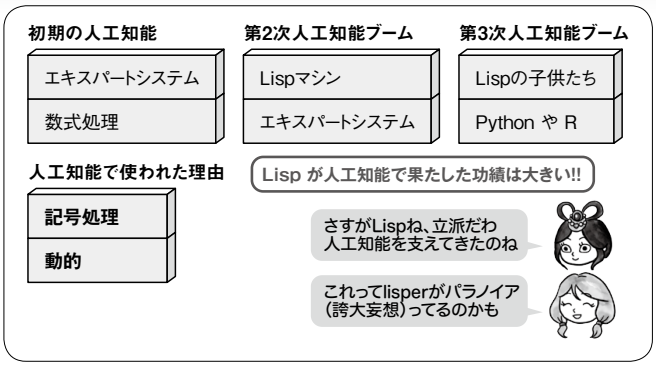
関数addには変数はなく、状態[注3]を持っていませんので、いつでもどこで実行しても同じ引数に対して同じ結果が返ってきます。このため、addを同時に並列に実行しても正しく結果を返します。面倒な同期を取る必要もありませんし、同期によるデッドロックも起こりません。純粋に数学の関数と同じ働きをします。一方、sumは同じ引数でもいつも結果が同じであるとは限りません。

関数型プログラミングの肝は副作用のないプログラムを作ることであり、高階関数やラムダ式[注4]、遅延評価[注5]などは便利な機能でありますが必須ではありません（しかし関数型プログラミングを自慢したいときはこれらを使ってみるのも手です）。

Lispが人工知能を作った

Lispは記号処理用言語として、初期の人工知能システムの実装で多く使われました（**図2**）。たとえば初期のエキスパートシステム[注6]であるMycin[注7]や、数式処理システムのMacsyma[注8]やREDUCE[注9]、初期の自然言語処理プログラムELIZA[注10]などがLispで開発されました。第2次人工知能ブームのときでもSymbolics[注11]やELIS[注12]などのLisp専用マシンが人工知能マシンとして作られ、その上で各種のエキスパートシステムが動作していました。

▼図2　Lispは人工知能の功労者



注3）関数の入力（引数）と出力（値）以外に何らかの変化をするものを状態と言います。たとえば、グローバル変数やローカル変数、ヒープ領域、ファイル、ネットワーク、画面表示、キー入力などがあります。

注4）高階関数とは、関数を呼び出すときの引数として、整数10や変数xなどと同じように、関数そのものを扱えます。また関数の戻り値として、関数を返すことです。ラムダ式とは関数型プログラミングの基本原理であるラムダ計算をするときに使う式で、これについては後述します。

注5）遅延評価とは、通常の関数の評価順である内側優先、左側優先で評価するのではなく、たとえば、その値が必要になるまで評価を遅延させる評価方法のことです。これにより、無駄な評価をしなくて済むようになります。どのように評価順序を決定するのかが評価戦略です。

注6）専門家（エキスパート）の判断を人工知能で行おうとするもの。

注7）1970年代にスタンフォード大学のブルース・ブキャナンとエドワード・ショートリッフェが開発した医療エキスパートシステム。

注8）Project MAC's SYmbolic MAnipulator。MITでカール・エンゲルマン、ウィリアム・A・マーチン、ジョエル・モーゼスにより1968年から開発されたMaclispによる数式処理システム。

注9）1960年代にアンソニー・C・ハーンによって開発された物理学用の代数計算数式処理システム。

注10）1960年代にMITのジョセフ・ワイゼンバウムによって開発された。

注11）MITの人工知能研究所から派生したコンピュータ製造企業。1980年にケンブリッジに設立され、Lisp実行に最適化されたマシンを設計製造していた。

注12）1985年に電電公社（現NTT）電気通信研究所で開発された国産Lispマシン。

▼リスト1　関数型プログラミングの例

```
●副作用がない関数（破壊的再代入がない関数）
(defun add (x y) (+ x y))  ; addは誰がいつでもどこで実行しても同じ値になる
                           ; これにより、並列に実行しても問題は起こらない

●副作用がある関数（破壊的再代入をしている関数）
(setf total 0)  ; 初期値代入
(defun sum (x) (setf total (+ x total)))  ; sumの値は同じ入力でもいつも違う値になる
; 変数 total は破壊的に再代入され、値が変わるために、実行順序によって結果が変わる
; この例はグローバル変数であり、これではリエントラントでもない
; ローカル変数への破壊的代入でも関数内部で副作用が生じる（関数外部は副作用なし）
```

しかしこの天下も長く続きませんでした。このエキスパートシステムの作成方法が確立されると、エンジン部は高速なC言語で作られるようになり、Lispはその地位から転落してしまいます。しかし第3次人工知能ブームになるとLispから影響を受けたPythonやRなどが、人工知能とビッグデータの立役者になってきました。このようにLispが人工知能に果たす役割は今もこれからもあり続けるでしょう。

Lispが人工知能で大活躍している理由は何でしょうか。それはLispでは記号が簡単に使えるからです。たとえば"I have a pen"という自然言語はLispでは(I have a pen)のようにシンボルのリスト(S式)として表せます。またルールベースのエキスパートシステムでもLispが大活躍していますが、これもLispでルールが簡単に記述できるからです。例として成績規則を考えた場合、それを**リスト2**のような動的なデータとしてLispのリストで簡単に表現できます。

そして前回(2017年5月号)の記事で紹介したように、リストを処理する豊富な関数が用意されています。まさにLispは記号処理のために作られた言語です。この記号処理で自然言語処理や数式処理、エキスパートシステムなどの人工知能システムが作られました。

しかしLispはこれだけの理由で人工知能に使われたわけではありません。別の強力な武器があります。それはLispの強力な動的機能[注13]です。これこそが人工知能用言語としての地位を築いた本当の理由です。当初の人工知能はどのように実装すれば動くのか、また効率的に動くのかがわかりませんでした。試行錯誤を繰り返しながら、記号処理の実験を繰り返して、初期の人工知能を作っていました。その過程ではプログラムやデータ構造を動的に変更したり拡張したりする機能が必須でした。いちいち再起動せずにインタプリタの実行中に関数が再定義できることは便利な機能でした。事実、静的な関数型言語であるMLなどでは人工知能にはあまり使われていませんでした。

注13) 動的機能とはプログラムの実行中に変更できる機能です。たとえば、プログラムの実行中に関数定義そのものを変えることができることを動的関数定義と言います。Lispには変数の生成やクラスの変更など各種の動的機能があります。一方、静的機能ではプログラムのコンパイル時などプログラムが静止しているときにのみ変更できる機能です。

▼リスト2　成績規則の記述例

```
(grading-rule
  ((< score 30) poor)
  ((and (>= score 30) (< score 50)) fail)
  ((and (>= score 50) (< score 70)) average)
  ((and (>= score 70) (< score 90)) good)
  ((>= score 90) excellent) )
```

Lispがオブジェクト指向の世界をおもしろくした

1970年代になると、Simulaなどで採用された抽象データタイプの概念からオブジェクト指向が萌芽してきましたが、エポックとなったのは1980年のSmalltalk-80[注14]の登場です。これにより、オブジェクト指向が一躍、脚光を浴びました。この時期、プログラミング言語の実装実験場であったLispでもMIT FlavorsやCommon Loopsなどの数多くのオブジェクトシステムが実装されました。そしてCLOS(Common Lisp Object System)がオブジェクト指向の新しい可能性に挑戦したものになりました(**図3**)。

CLOSはSmalltalkやJavaなどのクラスベースのオブジェクト指向ではなく、関数を拡張した総称関数[注15]をベースにした新しいオブジェクト指向です。これは関数型プログラミングの基本である関数をそのまま総称関数に拡張して、オブジェクト指向機能を導入したものです。ま

注14) ゼロックスのパロアルト研究所で開発され、SimulaやLisp、Logoの影響を受けた単一継承のオブジェクト指向言語。オブジェクト指向言語の発展に大きく寄与し、その後に続くJavaやC#などに大きな影響を与えた。

注15) Generic Function。関数の引数クラスと値クラスの組み合わせごとのメソッド関数をひとまとめにして総称した関数。(メソッド)関数が特定の引数クラスと値クラスの組み合わせの1個だけの関数であるのに対し、総称関数は(メソッド)関数を集めた関数。たとえば(defmethod add ((x number) (y number)) (+ x y))や(defmethod add ((x list) (y list)) (append x y))が各々の引数クラスの組み合わせのメソッド関数1個であるのに対し、(defgeneric add (x y))はこれらのメソッド関数をひとまとめにしたもの。

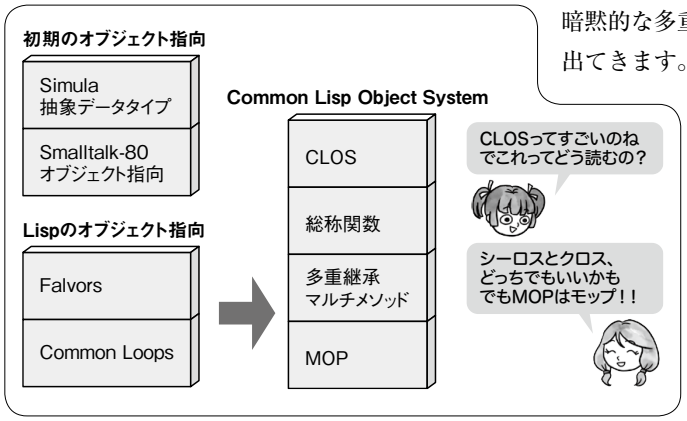
たCLOSは多重継承[注16]やマルチメソッド[注17]など革新的な機能をサポートし、さらにLispらしく動的にオブジェクトシステムを拡張[注18]できるようになっています。まさにLispがオブジェクト指向の世界をおもしろくしました。**リスト3**にCLOSのサンプルプログラムを紹介します。

注16) Multiple Inheritance。SmalltalkやJava、C#など親クラスが1個だけの単一継承を拡張して、複数のクラスを親クラスとして継承すること。たとえばx方向に伸縮可能なxウィンドウとy方向に伸縮可能なyウィンドウのクラスがあったとき、xとy方向に伸縮可能なxyウィンドウが、xウィンドウとyウィンドウの両方のクラスを継承すること。

注17) Multimethods。総称関数の中で適用可能な複数のメソッド関数から呼び出されるメソッド関数を選ぶこと。または適用可能な複数のメソッド関数があること。

注18) MOP（MetaObjectProtocol）とも言う。MOPとはインスタンス生成やクラス構造、継承のしくみなどのオブジェクト指向機能そのものを決めるものであり、MOPを定義することで自由に動的にオブジェクト指向機能を変更できる。

▼図3　Lispがオブジェクト指向をおもしろくした



CLOSの特徴には**注16**で紹介したxyウィンドウのような多重継承があります。C++のようなメンバ関数呼び出しのときに、明示的に親クラスを指定する多重継承ではなく、メソッド関数呼び出しのときに親クラスを指定しない暗黙的な多重継承です。もちろん、こちらの方が面倒がありません。CLOSは、どの親クラスを使うかを宣言することなしにプログラマにとって一番自然な順序で親クラスを見つけます（**リスト4**）。

このときに順序を単純な深さ優先で作ると、a、b、d、cの順序になります。しかし、これはaのローカルの順序a、b、c、dに反します。この順序に反しないように並べ替えをして、たとえばa、b、c、dの順序にします。この並べ替えをトポロジカルソートと呼びます。このように暗黙的な多重継承では面倒な問題がいろいろと出てきます。

Lispはプログラマに余計な負担をかけず、不自然な継承をしません。一方、Smalltalk-80から始まる単一継承のオブジェクト指向言語、たとえばJavaやC#、Rubyなどはスーパークラスを複数持つことはできず、プログラム開発で強い制限になります。はっきり言って手抜き工事です。

▼リスト3　総称関数のプログラム

```
●クラスpersonの定義、スロット変数（Javaではフィールド変数）にはnameとageを定義
(defclass person () (name age)) → #<STANDARD-CLASS PERSON>
(defclass food () (name prise)) → #<STANDARD-CLASS FOOD>
●クラスpersonとfoodの2個のクラスを引数とする総称関数eatを定義
(defmethod eat ((person person) (food food)) (list person food))
  → #<STANDARD-METHOD (#<STANDARD-CLASS PERSON> #<STANDARD-CLASS FOOD>)>
●シンボルpersonの値として、personクラスのインスタンスオブジェクトを代入
(setf person (make-instance 'person)) → #<PERSON アドレス>
(setf food (make-instance 'food)) → #<FOOD アドレス>
●総称関数eatを実行
(eat person food) → (#<PERSON アドレス> #<FOOD アドレス>)
```

▼リスト4　自然な順序

```
(defclass a (b c d) ()) ; aの親クラスとしてb、c、dを多重継承。継承順序はa、b、c、d
(defclass b (d) ())     ; bの親クラスとしてdを継承。継承順序はb、d
```

再帰プログラミングは自然

Lispは、CやJavaなどのように繰り返しの機能を持っていますので、それを中心としたプログラミングもできます。ですがLispと言えば、やはり再帰プログラミングです。再帰プログラミングは関数型プログラミングの基礎として、再帰関数(帰納関数)の概念を使っています。再帰関数とは、直接かまたは間接的に自分自身の関数呼び出しをすることです。ここで例として正の整数の加算関数**add**を、**1+**と**1-**を使って再帰で作ってみましょう。参考に繰り返しプログラミングも示します(**リスト5**)。

再帰プログラミングと、繰り返しプログラミングの記述能力は同等です。つまり、どちらかのプログラム手法は他方のプログラム手法で書くことができます。それでは、この再帰プログラミングと今までの繰り返し文を使ったプログラミングと、どちらが自然なプログラミングだと思うでしょうか。

手続き型プログラマ脳になっていると、繰り返しプログラミングが自然だと感じているかもしれません。でも待ってください。まずは例を見ていきましょう。再帰プログラムのadd関数では、**(1+ (add x (1- y)))**のように、加算とは**1-**して加算したものに**1+**したものであるという帰納的定義[注19]になっています。

一方、繰り返しプログラミングは**x**に**y**回**1+**するという手順しか示していません。つまり繰り返しプログラミングは、結局は手続きの方法を言っているだけで、どんなものを作りたいのかは言っていません。手順書だけ渡されて完成図がないのです。一方、再帰プログラミングは帰納的定義で何を作るかの定義をそのまま最初に書いています。

つまり再帰プログラミングは手順(How：手続き)でなく、モノ(What：定義)を書いています。手続き型プログラマ脳に侵されていない状態なら、繰り返しよりも再帰の方がとっつきやすいでしょう(**図4**)。事実、子供向けのプログラミング言語のScratchやそれに影響を与えたLogoなどは、再帰プログラミングが基本になっています。

再帰プログラミングが再帰中の引数や値の退避をするためにスタックを使う分、実は実行効率は落ちます。再帰プログラミングはスタックを大量に消費し、最悪、スタックオーバーフローのエラーが出ます[注20]。一方、繰り返しプログラミングはローカル変数にどんなに破壊的代入文を繰り返しても、ローカル変数をスタックに退避する必要がなく、スタックをまったく消費せずに高速に実行できます。

でも安心してください。再帰を末尾再帰[注21]にすれば、再帰プログラミングもスタックを消費せずに単純な繰り返し文と同等になります。最

注19) 帰納的定義とは今までの定義を使って新たに定義することです。たとえば、n - 1まで定義されているものを使って、nを定義することです。再帰関数は帰納関数とも呼ばれています。

注20) 再帰ではシステムスタックへ引数や値を格納して、スタック配列に対する破壊的代入をしています。もちろん、これらは副作用です。しかしLispやHaskellなどの処理系自身が行うことはガベージコレクションも含めて完全に正しく動作するものとして、外部に副作用を見せません。だから副作用はない、というのが関数型言語の言い草です。

注21) 最後に関数呼び出しが行われる関数(末尾関数)が自己再帰する関数を末尾再帰と言います。たとえば、**(defun fact (n) (if (<= n 1) 1 (* n (fact (1- n)))))**の再帰関数を、**(defun fact (n ret) (if (<= n 1) ret (fact (1- n) (* n ret))))**のようにfactを最後に再帰呼び出しするように変更すると、末尾再帰になる。

▼リスト5　加算関数addの、再帰プログラミングと繰り返しプログラミング

●再帰プログラミング

```
(defun add (x y)
  (if (= y 0)                    ; ベース項の判断で、yが0かどうかで判断
    x                            ; ベース項で、yが0のときはxを返す
    (1+ (add x (1- y))) ))       ; 再帰項で、addを自己再帰呼び出しをしている
```

●繰り返しプログラミング

```
(defun add (x y)
  (let ((s x))                                ; ローカル変数sで初期値はx
    (dotimes (i y s) (setf s (1+ s))) ))      ; 制御変数iの値が0からy-1まで繰り返す
```

近の関数型言語では自動的に末尾再帰の最適化を行うようになり、一方スタックも大量に高速にアクセスできるようになり、効率の面からも問題はなくなっています。本当です。

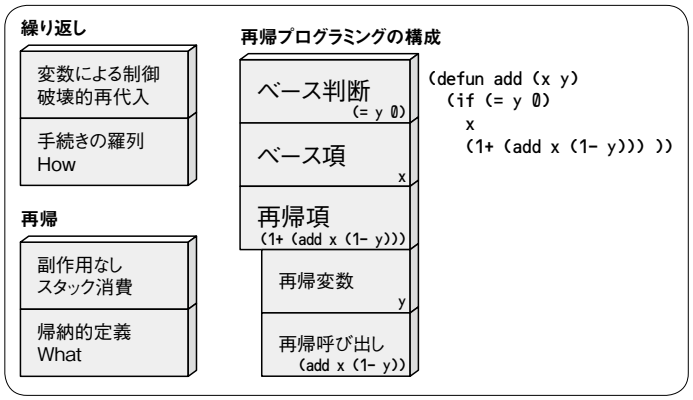
マクロで世界を広げる

Lispには強力なマクロ機能があります。一言で言って、マクロを使えば何でもできます。Lispのマクロが強力な理由は、マクロ定義そのものがプログラムになっていることです。言わば、メタプログラミング[注22]ができます。それではマクロの例を見ていくことにします(**リスト6**、**図5**)。

マクロ定義をこのようにLispプログラムで書くことができますが、多くはlistなどを使って、S式プログラムを構成することになります。そこで、いちいちlistなどで書かなくてもいいように、マクロ定義のときの便利な記法として、バッククォート記法があります(**リスト7**)。`(バッククォート)や,(カンマ)、@を見ないようにすれば、ほぼ展開形のように見えるので、前のマクロ定義と比較して、だいぶ見やすくなりました。

この強力なマクロを使って、自由にLispの構文を拡張できます。まさにプログラミング言語の実験場としてぴったりです。ぜひ、マクロでLispを拡張してみてください。

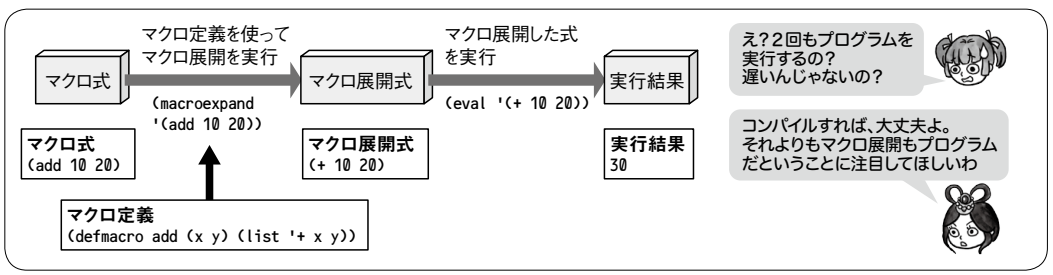
▼図4　再帰プログラミングでGo!



▼リスト6　マクロの例

```
; defmacroではマクロ展開する方法をLispでプログラミングする
; ここでは (add 10 20)を (+ 10 20)になるように、listを使ってプログラムしている
(defmacro add (x y) (list '+ x y))
; マクロを展開する(関数macroexpandはマクロ式をマクロ定義に従って展開する関数)
(macroexpand '(add 10 20)) → (+ 10 20)
; マクロを実行する
; マクロ式のときはそれを展開して、さらに展開したものを実行する(2回プログラムが実行される)
(add 10 20) → (+ 10 20) → 30
```

▼図5　マクロ定義、展開、実行



▼リスト7　バッククォート記法のマクロの例

```
(defmacro add (x y) `(+ ,x ,y))          ; `(+ ,x ,y)が(list '+ x y)の簡略記法
(macroexpand '(add 10 20)) → (+ 10 20)   ; , は評価(引数はその値に、S式は実行する)
(defmacro my-list (&rest x) `(list ,@x)) ; `(list ,@x)が(cons 'list x)の簡略記法。@はconsでつなぐ
(macroexpand '(my-list 1 2 3 4)) → (list 1 2 3 4) ;&restは以降の引数をリストにするもの
```

注22)メタプログラミングとはプログラムを生成するプログラミング。Lispでは動的にメタプログラミングができるが、マクロではメタプログラミングを言語仕様として持っている。

ラムダ式で苦労する

ここでは元祖ラムダ[注23]式を使ったプログラミングを見ていきましょう。JavaやC#などではこのラムダ式が後付けで採用されましたから、知っている人もいるかと思います。ラムダ式は一言で言えば、関数定義時の環境を引き連れた名前を持たない関数(匿名関数)のことです(**図6**)。このラムダ式はラムダ計算[注24]を原理にして動作します。ラムダ計算は変数束縛と関数適用などに関するたった3個しか規則がない簡単なものです。

このラムダ式はどんな役に立つのでしょうか。まずラムダ式の身近な利点としては、関数名を付ける苦労から開放されることがあります。一時的に使うだけの関数にわざわざ名前を付けるのは面倒で、そのコストも馬鹿にできません。次にラムダ式の本質的な利点とは、ラムダ式(匿名関数)そのものをデータとして扱えることです。ラムダ式をデータとして、そのときの環境を持ち運んで、別の関数の引数にしたり、関数の値として返したり、変数に代入したりすることができます。この考えは関数型プログラミングの節で紹介した高階関数に使えることになります。第1回で紹介したLispの最大の特徴である「プログラム＝データ」という考えも、このラムダ式の考えから来たものです。

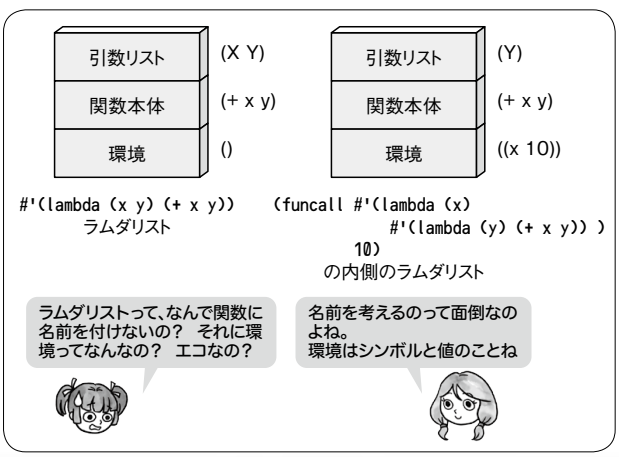
ここまでは、ラムダ式の薔薇(ばら)色の利点を見てきましたが、面倒な点もあります。関数をデータとして扱うときに、変数束縛の環境(どの変数がどんな値に束縛(代入)されているか)をどのように効率的に作るかは、Lisp処理系の実装者の腕にかかってきます(苦労します)。

以下、環境がない場合のラムダ式の例を見ていきます。このラムダ式の実装を**図7**と**リスト8**に紹介します。**リスト8**のプログラムの理解は面倒かもしれませんが、**図7**の概要を括弧だらけで書いているだけです。安心してください。

このラムダ式は前述したようにLispのファーストクラスオブジェクト(つまり普通に読み書き

▼図6　ラムダ式

▼図7　ラムダリストの実装例



注23) アロンゾ・チャーチが1941年に提唱したラムダ計算は、なぜラムダ(λ)だったのでしょうか。有名な数学書である「数学原理」では束縛変数に「^(カレット)」を使っていました。チャーチはこれをΛ(大文字のラムダ)で表しましたが、他と混同しやすいので、小文字のλにしました。チャーチの学生だったマッカーシーがLispにそのλをlambdaとして採用したのがラムダ式の始まりです。

注24) ラムダ計算の規則には変数名を変更するα-変換と、関数適用するβ-変換、そして関数の等価性判断のη-変換の3個しかありません。関数適用のβ変換がラムダ計算の中心になります。またラムダ計算では変数が束縛された変数(ローカル変数に相当)と束縛されずに自由に出現する変数(グローバル変数)があります。またラムダ計算で使うラムダ項はλx.fxyのように記述し、xが束縛変数、yが自由変数になります。このラムダ計算については、拙著『はじめてのLisp関数型プログラミング』(技術評論社、ISBN978-4-7741-8035-9)を参照してください。

できる値)として扱えます。**リスト8**中の(2)や(3)のように関数の引数や値にすることもでき、ラムダ式は値として自由に持ち運びができます。これが関数型プログラミングの、そしてLispのいいところです。

次はいよいよ環境を持ったラムダ式(Lispではクロージャ(関数閉包)で実装)の説明になります。Lispを勉強すると誰もが一度は聞く駄洒落に「クロージャは苦労するんじゃー」がありますが安心してください。本当です。

クロージャは環境をラムダ式に閉じ込めます。ここで環境とはラムダ計算の変数束縛のことです。一般のプログラミング言語的に言えば、ローカル変数とその値です。

リスト9をご覧ください。(1)の例では**(funcall fun1 10)**を実行することで、ローカル変数xとその値の10を閉じ込めているクロージャを生成します。そのクロージャ(fun2に代入されている)を実行すると、閉じ込めた環境からxの値を検索して、10を返します。このようにクロージャにより環境を持ち運びできます。そのクロージャが生きている間、環境も生き続けることになります。

このように定義時の環境を含んだ関数(クロージャ)を自由に持ち運べるので、プログラムも柔軟で拡張性が高く、動的なプログラミングができます。この有用性が認められて、JavaやC#などの言語にもラムダ式が採用されてきています。ラムダ式やクロージャを怖がらずに気楽に使ってください。

最後に

今回まではLispの使い方を見てきましたが、次回はこのLisp処理系自身の作り方を見ていくことでさらにLispを深く知ることにします。SD

▼リスト8　ラムダ式

```
● (1) ラムダ式の表現
; このラムダ式は引数xとy(束縛変数xとy)で、(+ x y)の加算を実行する
; (function (lambda ...))が正式であるが、それを簡略化した#'(lamda ...)やマクロlambdaも使える
#'(lambda (x y) (+ x y))
● (2) ラムダ式の実行とラムダ式の引数、データとしてのラムダ式
; ラムダ式はfuncallで以下のように実行できます。
; funcallの第1引数にはラムダ式、第2引数以降はラムダ式の引数
(funcall #'(lambda (x y) (+ x y)) 10 20) → 30
(setf fun #'(lambda (x y) (* x y))   ; シンボルfunの値にラムダ式を代入する
(funcall fun 10 20) → 200          ; シンボルに代入されたラムダ式を実行する
● (3) 関数の値になるラムダ式
; ラムダ式を返す関数を作ります
(defun fun () #'(lambda (x y) (+ x y)))  ; 加算のラムダ式を返す関数
(fun) → #<FUNCTION :LAMBDA (X Y) (+ X Y)>
(funcall (fun) 10 20) → 30
```

▼リスト9　クロージャの例

```
● (1) 環境を閉じ込めたラムダ式(クロージャ)
; 自由変数xを返すラムダ式 #'(lambda () x)を値にするラムダ式
(setf fun1 #'(lambda (x) #'(lambda () x)))
; 上記の外側のラムダ式を実行する
(setf fun2 (funcall fun1 10)) → #<FUNCTION :LAMBDA NIL X>
; ローカル変数xとその値が10である環境を閉じ込めたクロージャを返す
; このクロージャを実行する
(funcall fun2) → 10   ; クロージャで閉じ込めていた環境を使ってxの値を返す
● (2) 局所束縛letによるクロージャ
; letによりローカル変数xを10に束縛している。(1)と同じでlambdaの略記法として捉える
(setf fun2 (let ((x 10)) (lambda () x))) → #<FUNCTION :LAMBDA NIL X>
(funcall fun2) → 10
```

RDBアンチパターン

PostgreSQLとMySQLの失敗と対策

Author 曽根 壮大（そね たけとも） ㈱はてな Twitter @soudai1025

第4回 効かないINDEX

本連載では、開発の現場で発生しやすいリレーショナルデータベース（RDB）全般の問題をRDBアンチパターンとして紹介していきます。今回のアンチパターンの主人公は、INDEXを使っているのになぜかクエリが遅いという問題にぶつかった、データベース管理者。

前回は「やりすぎたJOIN」と題して、RDBだからこそ陥るJOINの罠を説明しました。第4回にあたる今回は「効かないINDEX」という題名で、RDBのチューニングの際に必ず利用するINDEXについてお話します。

効かないINDEX

知っているようで意外と知らないINDEX。実はRDBMSによってそのしくみや種類は異なります。INDEXなしではRDBの高速化は語れませんので、今回はINDEXを深掘りします。

事の始まり

DBA（データベース管理者）のTさんはスロークエリログを見ながら悩んでいる最中。

Tさん：なんでだろう……急にこのクエリが遅くなってる。手元で試してみてもINDEXを使ってくれてるのに。

そこへ同僚のKさんが助け舟を出してくれた。

Kさん：本番と同じデータでちゃんとチェックした？　データが大き過ぎるなら、バックアップ用にレプリケーションしているスレーブで試してごらん。

Tさん：アドバイスありがとう。試してみるよ！

さっそく試してみるTさん。

Tさん：なるほど……、同じクエリでもWHERE句に指定する値によってINDEXが効いたり効かなかったりするぞ。あれ？　同じデータ（**リスト1**）なのにステージングでも再現しない実行計画がある。これはもう少し、INDEXについて勉強する必要があるぞ。

INDEXの役割

INDEX（索引）とはテーブルからデータを高速

▼リスト1　該当のテーブルDDL

```
CREATE TABLE `users` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `gender` tinyint(1) NOT NULL COMMENT
 '1 =  男性, 2 = 女性',
  `age` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `index_name` (`name`),
  KEY `index_age` (`age`),
  KEY `index_gender` (`gender`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

に取り出すためのRDBMSのしくみです。

INDEXを適切に作成し、さらにそれを検索時に利用することでクエリの高速化が実現されます。

INDEXはRDBMSによって実装方法が異なります。同じOSSでも、MySQLとPostgreSQLでは実装に違いがあります。

BTree INDEXのしくみ

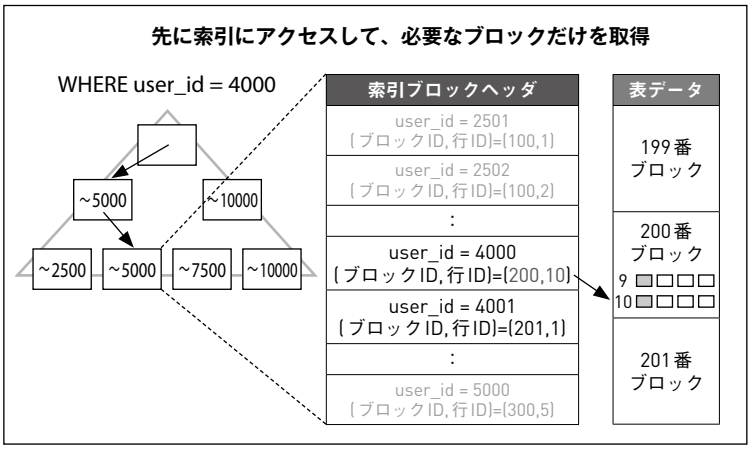
よくINDEXを使いましょうと一般的に言われていますが、実はINDEXにはいろいろな種類があります。その中でも、一般的にRDBMSで利用されているINDEXがBTree INDEXです。MySQLでもPostgreSQLでも、多少の細かい実装は違えど考え方は一緒で、みなさんが日々使うINDEXはBTree INDEXと言って良いでしょう。ですので、今回はこのBTree INDEXのみを扱います。以降、本文中のINDEXはBTree INDEXのことを指すとします。

それではさっそくBTree INDEXの大まかなしくみですが、**図1**のとおりです。この例では、

```
SELECT * FROM users WHERE user_id = 4000
```

の結果を取得するために、INDEXを使わないテーブルスキャンでは200ブロックを読み込む必要があります(**図2**)。それに対し、INDEXを利用すると4ブロックのアクセスのみで済みます(**図3**)。単純に比較しても50倍の効率ですね！

▼図1　BTree INDEXの構造(PostgreSQLの例)



このとおり、INDEXを利用することでテーブルスキャンに比べ高速にデータを検索できます。

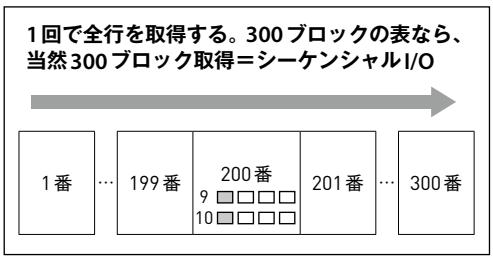
今回のアンチパターン

ここまでINDEXのしくみについて説明しました。INDEXを有効に使うことで高速にデータを検索できます。今回のアンチパターンには、次のような問題があります。

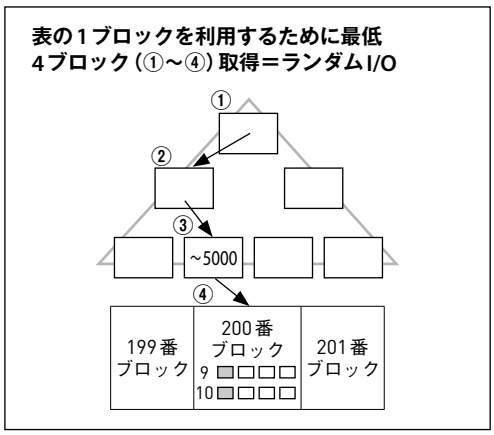
- INDEXのしくみと、RDBMSがいかにそれを利用しているかを知らない
- INDEXを利用するにはテーブル設計が不適切

INDEXの特性を交えて説明していきます。

▼図2　テーブルを直接読むコスト(テーブルスキャン)



▼図3　INDEXを利用するときのコスト



設定したINDEXが効かない（使われない）ケース

次のようなケースでは、RDBMSはINDEXを使ってくれません。

検索結果が多い、全体の件数が少ない

図3では、INDEXを使うと1つのデータを取り出すために4ブロックにアクセスしています。1行を取り出すだけであれば、「1行×4ブロック」ですのでコストは4です。100行取り出すのであれば「100行×4ブロック」ですのでコストは400です。

テーブルスキャンの場合は「テーブルの行数×1ブロック」がコストです。もしテーブルが200行しかない場合は、「200行×1ブロック」ですのでコストが200です。

極端な例を言いますと、1万行のテーブルから9,999行を取り出すような場合、INDEXを使うよりもテーブルスキャンのほうが高速です。また、10行の中の5行を検索するような場合も、INDEXを利用した「5行×4ブロック＝20コスト」よりも「10行×1ブロック＝10コスト」のテーブルスキャンのほうが高速です。

INDEXを利用するためには次の2つの条件が必要です。

① 検索結果がテーブル全体の20%未満

一般的な実務レベルでは10%未満を指標にするのが良い

② 検索対象のテーブルが十分に大きい

数万～数十万行が目安。1,000行程度のテーブルの場合はINDEXを参照するよりもテーブルスキャンが効率的なケースが多いため、INDEXは利用されにくい。たとえば都道府県マスタのように47行しかないテーブルの検索では、INDEXが利用されないケースが大半

ここで注意すべきこととして、データの比率は時間とともに変わります。たとえばユーザの世代別に集計する必要があった場合に、世代の比率によってINDEXが有効活用できる／できないが決まります。次のような10万件の会員データがあったとします。

10代が10%／20代が50%／30代が10%／40代以上が30%

この場合に30代を抽出するクエリ、

```
SELECT * FROM users WHERE age BETWEEN 30
AND 39;
```

はINDEXが有効に利用される可能性が高いです。逆に20代は全体の50%ですので、INDEXは利用されないでしょう。

ではここで、このサービスの会員にまったく流動がないまま10年が経った場合はどうでしょうか？　当たり前ですが、現在の20代がそのまま30代になります。そのため、現状ではINDEXを利用している上記クエリが時間経過とともにある日、突然INDEXを利用しなくなります。これがまさに、冒頭のTさんのような例で見られるケースです。

条件にその列を使っていない

INDEXは、検索対象の列がWHERE句やJOINの際のON句などで利用されていない場合、利用できません。一見すると当たり前に見えますが次のような例が該当します。

・インデックスが利用されない例

```
SELECT * FROM users WHERE age * 10 > 100;
```

この例では、INDEXをage列に対して設定している場合でもINDEXは使われません。検索の対象はage * 10の計算結果となるため、すべての行に対して計算・比較する必要が出てくるからです。この場合は次のように変更することでINDEXを利用するクエリとなります。

・インデックスが利用される例

```
SELECT * FROM users WHERE age > 100/10
```

類似例として、対象の列を関数の引数に指定している場合があります。

・インデックスが利用されない例(MySQL)

```
SELECT * FROM `原稿` WHERE
UNIX_TIMESTAMP(〆切期日) < $unix_timestamp;
```

・インデックスが利用される例(MySQL)

```
SELECT * FROM `原稿` WHERE
`〆切期日` < FROM_UNIXTIME($unix_timestamp);
```

こちらの例も同様です。INDEXは関数の結果を持っているわけではないので、すべての行に対して関数を実行する必要があります。そこで、関数の結果をINDEXを設定した列と比較するようにしています。

PostgreSQLには式INDEXという機能があります。これは関数などの式の結果をINDEXにするという機能で、前述のUNIX_TIMESTAMP()のような結果をINDEXにできます。そのため、どうしても関数を利用する必要があるケースでも高速に検索できます。たとえば次のようなクエリでも、式INDEXを利用できます。

```
SELECT * FROM users WHERE SUBSTR(name,
0, 2) = '曽根';
```

✤ カーディナリティの低い列に対する検索

「列に格納されるデータの値にどのくらいの種類があるのか?」をカーディナリティと言います。種類が多い、つまり重複が少ないデータはカーディナリティが高いことになります。たとえばシーケンシャルなidは、各行ごとに新たな値が指定される場合は重複がなく、データの種類が多くあるのでカーディナリティが高いです。

逆にカーディナリティが低い例は、今回のアンチパターンですと「性別」になります。**リスト1**のgender列は、1 = **男性**, 2 = **女性**と2種類のデータしかないため、データに重複が多くなります。このようなカーディナリティが低い列に対して絞り込む場合は検索結果が多くなりやすいため、INDEXをうまく利用することが難しいのです。

たとえば男女比が1:1であれば、genderに対する結果が常にテーブル全体の50%になるため、WHERE gender = 1などの検索よりも、別の列に設定されたINDEXを利用したほうが有効となります。しかし男女比が1:99の場合、WHERE gender = 2などの検索は99%が検索対象となるため、INDEXが有効活用できません。逆にWHERE gender = 1の場合は1%が検索対象になるため、有効にINDEXを利用できます。

✤ あいまいな検索

RDBMSの検索にLIKE検索があります。パターンとしては**表1**の3種類があります。このうち、標準でINDEXを利用するケースは前方一致のみです。そのため、後方一致でINDEXを利用したい場合は、reverse()などの関数で対象の列をひっくり返して別の列に保存したり、PostgreSQLの式INDEXを利用したりする必要があります。

部分一致でINDEXを利用したい場合は、全文検索インデックスなどを利用する必要があります。こちらはRDBMSや使う拡張などによっても大きく特徴が変わりますので、今回はツールの紹介までにしたいと思います(**表2**)。

▼表2　全文検索インデックス

RDBMS	ツール名	URL
MySQL	Mroonga	http://mroonga.org/ja/blog/
PostgreSQL	PGroonga	https://pgroonga.github.io/ja/
	pg_bigm	http://pgbigm.osdn.jp/

▼表1　LIKE検索

種類	概要	例
前方一致	検索対象の中で指定した文字列から始まる単語に一致	WHERE LIKE 'TEST%'と指定するとTEST 01、TEST 02などにヒット
後方一致	検索対象の中で指定した文字列で終わる単語に一致	WHERE LIKE '%TEST'と指定すると01TEST、02TESTなどにヒット
部分一致	検索対象の中で指定した文字列が含まれる単語に一致	WHERE LIKE '%TEST%'と指定するとTEST 01、01TESTなどTESTを含むすべてにヒット

✤ 統計情報と実際のテーブルのデータ状態とで乖離（かいり）がある場合

INDEXを利用するかどうかは、クエリの実行時にオプティマイザが判断して決めています。このときの判断材料となるのが統計情報です。統計情報は定期的にテーブルから一定数のサンプリングを行い、それをもとに作られます。

ただし、「実際のデータ分布から乖離した統計情報」が作られることがあります。次のような場合にそれは起こります。

❶ サンプリングの前に大量のデータ更新が行われた（例：バッチ処理でデータを大量に追加した、範囲の広いUPDATEを行ったなど）
❷ サンプリングで偏ったデータを収集した

❷は、データとクエリはまったく同じなのに、本番とステージングで実行計画が違う場合の原因になることもあります。

これらのような場合は統計情報の更新を行いましょう。しかし、統計情報の更新によって実行計画が変動することが好ましくない場合もあります。そのようなときのために、利用するINDEXや統計情報を固定する手法があります。基本的にはオプティマイザに任せることが適切なクエリのほうが多く、この手法はあくまで飛び道具ですので、参考ページの紹介までにしたいと思います（**表3**）。

このアンチパターンのポイント

今回のアンチパターンは「INDEXの特性を知らない」ことで発生しました。また、仮にINDEXのしくみを知っていても、データの変化も含めて適切に設計しなければ将来的にINDEXが利用されなくなることがあることもわかりました。今回のアンチパターンの対策のポイントをまとめますと、次のとおりです。

- INDEX（とくにBTree INDEX）の特性をしっかりと把握して適切なINDEXを設定する
- INDEXを利用できるクエリを実行する
- INDEXを活用できるテーブル設計をする
- スロークエリログやデータの状態などをしっかりとモニタリングする

RDBMSは「今のデータ情報をサンプリングした統計情報」をもとにオプティマイザが判断しています。そのため未来のデータについては判断できず、そこも含めていかにテーブルを設計するかがエンジニアの腕の見せ所となります。今回の例は実務でもよく見られる例であり、設計時にどこまで考慮するか？——という判断も難しいため、継続的にデータの中身をモニタリングすることが大切になります。

次回のRDBアンチパターン

今回のRDBアンチパターンはいかがでしたでしょうか？　RDBMSを有効に使うにはINDEXが必要不可欠です。しかしINDEXも万能ではなく、特性をしっかりと理解することが重要です。

次回は、そんなINDEXの宿敵とも言えるフラグについてのアンチパターンです。このアンチパターンは、みなさんも現場で見る機会があるのでは？　次回の「フラグの闇」もお楽しみに！ SD

▼表3　インデックス・統計情報の固定方法

RDBMS	参考ページ・ツール名	URL
MySQL	公式ページ「インデックスヒントの構文」	https://dev.mysql.com/doc/refman/5.6/ja/index-hints.html
PostgreSQL	pg_hint_plan（PostgreSQLは標準ではヒント句がないため、拡張として追加する必要がある）	https://github.com/ossc-db/pg_hint_plan
	pg_dbms_stats（統計情報を管理するためのツール）	https://github.com/ossc-db/pg_dbms_stats

ROB TIPS インデックスショットガン

今回はINDEXを適切に有効活用できていないアンチパターンでしたが、名著『SQLアンチパターン』[注A]に「インデックスショットガン」というアンチパターンが出てきます。これは、「闇雲にINDEXを設定しまくる」というアンチパターンです。本編では触れませんでしたが、INDEXを設定することでINSERT/UPDATE/DELETEが遅くなるという問題があります。複雑な複合インデックス(複数の列に対するINDEX)を設定し過ぎると、オプティマイザが不適切なINDEXを選ぶことがあります。

これらについて、『SQLアンチパターン』の中では「MENTORの原則」に基づいて対応しましょうと書いてあります(**表A**)。この点は今回のアンチパターン「効かないINDEX」でもまったく同様です。

またINDEXは一般的に、追加よりも削除のほうが難しいです。なぜならば「このINDEXを外したことによるパフォーマンス遅延」は予測しづらく、リスキーである反面、INDEXを作りっぱなしするリスクはそれよりも比較的少ない場合が多いからです。

しかし、INDEXはテーブルデータを効率良く保存した実体のあるデータですので、ディスク容量も増えますし、作り過ぎは当然良くありません。今回のアンチパターンを見直し、INDEXを設計する際には、併せて次のような問いかけを自分にすることも大事です。

・このテーブルは1年後、3年後、5年後、何行くらいになるだろうか

データが少ないならINDEXは不要ですし、場合によっては後から追加することもできる。データ内容によって将来的にINDEXが効かなくなることが想定される場合はテーブル構造から見直す必要がある

・このINDEXは複合INDEXでまとめる、または単一のINDEXで十分絞りこめるのではないだろうか

「不要なINDEXが多いのではないか？」「代替できるINDEXがあるのではないか？」と一度振り返ることはインデックスショットガンを防ぐために非常に有効

・今このINDEXを張るべきか

INDEXはデータ傾向によっては活用できない可能性があるので、新規サービスなどでデータ傾向が見えない場合はリリース後の実際の傾向を見て判断することも大切

最後に筆者の経験則ですが、データベースの問題にはスケールアップすることで解決されるものが多いです。クラウドサービスが主流になりつつある昨今、スケールアップも大切な選択肢の1つです。

INDEXを使いこなすには、先月も紹介しましたが@yoku0825さんのスライド[注B]が非常にわかりやすいのでお勧めです。BTree INDEXにフォーカスした『SQLパフォーマンス詳解』[注C]というすばらしい書籍もあります(同じ内容をWebサイト[注D]で読むこともできます)。

MySQLやPostgreSQLなどのRDBMSにこだわらず、みなさんこの機会にぜひ、一読していただければと思います。

▼表A　MENTORの原則

項目	詳細
Mesure(測定)	スロークエリログやDBのパフォーマンスなどをモニタリング
Explain(解析)	実行計画を見てクエリが遅くなっている原因を追求
Nominate(指名)	ボトルネックの原因(インデックス未定義など)を特定
Test(試験)	ボトルネック改善(インデックス追加など)を実施し、処理時間を測定。改善後の全体的なパフォーマンスを確認
Optimize(最適化)	DBパラメータの最適化を定期的に実施し、インデックスがキャッシュメモリに載るように最適化
Rebuild(再構築)	統計情報やインデックスを定期的に再構築

注A) Bill Karwin 著、和田 卓人、和田 省二 監訳、児島 修 訳、オライリー・ジャパン発行、2013年
URL https://www.oreilly.co.jp/books/9784873115894/
注B)「WHERE狙いのキー、ORDER BY狙いのキー」 URL https://www.slideshare.net/yoku0825/whereorder-by
注C) Markus Winand 著／発行、松浦 隼人 訳、2015年 URL http://sql-performance-explained.jp
注D) URL http://use-the-index-luke.com/ja/sql/preface

RDB性能トラブルバスターズ奮闘記

原案 生島 勘富(いくしま さだよし) info@g1sys.co.jp ㈱ジーワンシステム
構成・文 開米 瑞浩(かいまい みずひろ) イラスト フクモトミホ

最終回 APIファースト・メソッドがプログラマを救う!

RDBやSQLにまつわる性能問題の原因を、コードの書き方、システム設計、開発体制とさまざまな側面から検証してきた本連載も、いよいよ今回で最終回。これまでに取り上げた内容を振り返り、要点を整理します。

登場人物紹介

生島氏
DBコンサルタント。性能トラブルの応援のため大道君の会社に来た。

大道君
浪速システムズの新米エンジニア。素直さとヤル気が取り柄。

五代氏
大道君の上司。プロジェクトリーダーでもある。

SQLはもっとも簡単なプログラム言語である

ジーワンシステムの生島です。この連載もついに最終回となりました。

「ええっ！　終わりでっか？　そろそろ本気出して活躍しよか思てましたのに！」

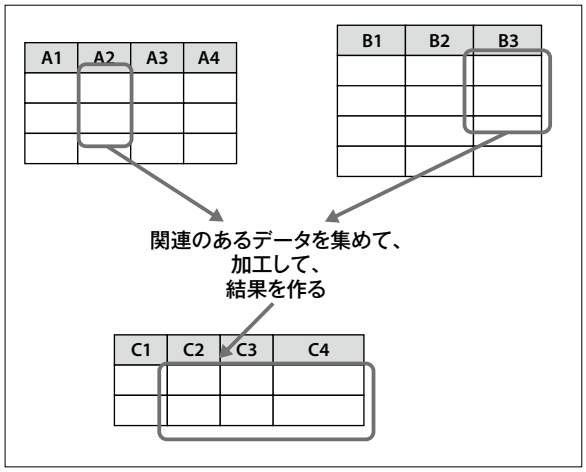
五代さんがなにやら慌てていますが、本当に最終回なので、今回は総まとめとしてこれまでの内容を振り返りながら、SQLにかかわるエンジニアに私がもっとも伝えたいメッセージを記すことにします。

1点目は「**SQLはもっとも簡単なプログラム言語である**」ということです。連載第14回[注1]で触れたように、普段Excelで事務作業をしている派遣社員さんにSQLを学ばせると、3日もあれば十分使いこなすようになります。これは「表形式のデータ」を見て、探して、加工するという仕事に日ごろから慣れているために「SQLが対象としている世界のイメージがある」からです。

図1のように複数の表の中の関連するデータの「集合」を切り出し、加工して、結果をまた別な表にしたり、もとの表に書き戻したりという操作をSQLなら非常に簡単にできます。この件は本連載の第2回や第3回[注2]で詳しく触れてあります。この種の処理を手続き型言語で書こうとするといくつもの制御変数を使って何回もループ処理を回さなければいけませんが、SQLはまさにそのようなプログラミングを不要にすることを目指して設計された言語なのです。

「そうですね、ほんと、集合的な操作をしてるんだという目で見るようになったら、SQLって

▼図1　SQLは表形式のデータ操作を簡単に行うことが可能



注1) 第14回：本誌2017年4月号。
注2) 第2回：本誌2016年4月号、第3回：2016年5月号。

実は簡単なんだということがわかるようになりました」と、大道君。

実はビジネスロジックもSQLのほうが書きやすい

「表形式のデータ」は事務処理でもっともよく使うデータ形式です。それを簡単に操作できるということは、ビジネスロジックもSQLのほうが記述しやすいということでもあります。

たとえば、「AかつBの場合はCの処理、AかつDの場合はEの処理……」のように場合分けをして処理するケースが業務システムでは頻繁にあります。これをストレートに実装しようとすると手続き型言語でif ～ then ～ elseの入れ子を作ることになりがちですが、実はこのような処理もSQLのほうが簡単に書けることが多いのです。

SQLではCASE式という機能によってきめ細かな場合分け処理を記述することができます。この件は連載の第7回[注3]や第14回で詳しく触れています。もちろん、ウィンドウ・システムを操作してUIを作ることはできませんが、UIに表示するデータを用意する処理についてはSQLのほうが簡単に書けます。

ビジネスロジックはSQLで書くべきである

したがって「**ビジネスロジックもSQLで書くべきである**」、これがメッセージの2点目です。ここでいうビジネスロジックとは、注文数を集計したり料金を計算したりといった処理のことを言います。

ビジネスロジックは場合分けを伴うことが多く、SQL文が複雑化することを嫌って手続き型言語で実装している例が非常に多いですが、極力DB側に移してSQLで処理するように考えてください。それは前述のように実は簡単に書けるということのほかに、仕様書が不要になる、性能も上がる、これらを通じて開発／運用のコストが抑えられる、といったことが理由です。

アルゴリズムを表す仕様書は不要になる

「仕様書が不要になる」とはどういうことでしょう？　仕様書と言ってもいろいろありますが、ここでいう仕様書は、「手続き型言語で実装するためのアルゴリズムを表した文書」のことです。**リスト1**がその例で、情報処理技術者試験でも定番のコントロールブレークと言われる処理を自然言語で仕様書化したものですが、いかにも「アルゴリズム」を表していることがおわかりでしょう。「顧客数を県別に集計する」という単純な処理ならやりたいことがそれだけで明確なのでわざわざ自然言語でこんなものを書くことはないでしょうが、実際に私が火消しに入る現場ではこんなイメージのもっと複雑な「自然言

注3）第7回：本誌2016年9月号。

▼リスト1　顧客数を県別に集計する処理

```
仕様書（自然言語でアルゴリズムを表現）だと……
 1. 顧客名簿を県別にソートする
 2. $kensu（件数）をゼロクリア
 3. 以下、ループ
     3.1 $kensuを1件加算
     3.2 次行の県名が現在行と変わっているか、
       最終行ならば、
         3.2.1 現在行の県名と$kensuを出力
         3.2.2 $kensuをゼロクリア
     3.3 最終行ならばループを抜ける

SQLで書くとたった1行
 SELECT 県名,count(*) FROM 顧客 GROUP BY 県名;
```

語でアルゴリズムを書いた」仕様書をよく見かけます。しかしSQLなら「アルゴリズム」を書く必要がありません。

「それも初めて聞いたときは意味がわかりませんでしたけど、よくよく考えたら、ああ、なるほどと思いましたね」

「そうやね～」と、大道君と五代さんがそろって言ってくれるのはうれしいです。

通常、プログラマは**図2**でいう「要求」を実現するために、それぞれの「言語」で「アルゴリズム」を記述します。これがプログラムです。「プログラム」を書く前に「アルゴリズム」を簡略に表現したものを「詳細仕様書」や「プログラム仕様書」の名前で書いている例が多いのですが、そもそも本質的に必要なのは「要求」であってアルゴリズムではありません。

その点、SQLが表しているのは「要求」です。**リスト1**の最後にSQLで実装した例を載せておきましたが、それを**図2**と見比べてもらえば「SQL文は要求を表現している」ことがわかるでしょう。

SQLもDBMSの内部ではループ処理に変換されて実行されるのですが、それはDBMSが自動的にやってくれるのでプログラマが手をかける必要がありません。つまり、SQLを使うなら**図2**でいう「プログラム」は不要、したがってその部分を表す仕様書も不要というわけです。

当然、仕様書を書く工数は不要になりますし、プログラマが書くコード自体も減ってシンプルなものになるためバグが入りにくいという意味でも開発コストを減らすことができます。その実例については第2回や第14回で触れています。

性能改善のためにもSQLの活用が効果的

加えて、要求を手続き型言語で実装する際に頻出する「ループ」は性能を悪化させる主要因でもあります。性能を改善するためにはループ処理は極力DB側に閉じた世界で完結させるべきです。その場合、**リスト1**のSELECT文の例でもわかるようにSQL文の中ではループは出てきません。そうして手続き型言語では極力ループを書かず、SQL文を呼び出す回数やDB→AP間のデータ転送量を最大限減らすことが求められます。

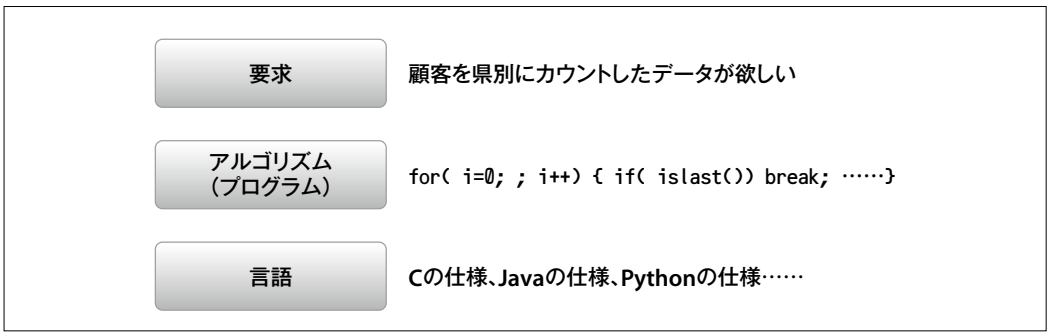
この件はいわゆる「ぐるぐる系」問題として連載初期に触れました（第1[注4]～3回）。性能が改善するとその分スペックの低いサーバで処理できるため、運用コストダウンにもなります。結局、開発／運用の双方でコストダウンになるため「ビジネスロジックはSQLで書くべき」なのです。

ただし熟練するには専門的な勉強と経験が必要

しかしそんなにメリットがあるのに、実際に十分なレベルで「ビジネスロジックまでSQLで」書いている開発現場はあまりありません。なぜないのでしょうか？　その理由の1つとしてありそうなのが、SQLを理解するために必要な「集

注4） 第1回：本誌2016年3月号。

▼図2　アルゴリズムは要求と言語の間をつなぐもの



合指向の考え方」が、手続き型言語を学んだときの感覚とギャップがあり過ぎて「よくわからん」となってしまっているケースです。

ほとんどの方が最初に学ぶプログラム言語は手続き型をベースにしていて、学習の主要なテーマはループや条件分岐という「制御構造を理解」し、それを自分で使って「アルゴリズムを組み立てる」ことでした。つまり、**図2**でいう「アルゴリズム」以下の部分を扱うのがプログラマの仕事でした。C、Java、Python、Ruby、PHPなど主要な手続き型系言語はいずれも基本的に同じしくみの制御構造構文を持っていて、だからこそ1つの言語を覚えれば2つめ以降を覚えるのは楽だったはずです。しかしそれがSQLについては通用しません。

「そうなんですよね、実際、SQLはJavaとは感覚が違い過ぎて、Javaの延長みたいに考えている間はダメでした……」

SQLは**図2**の「要求」部分を表現する言語であり、しかも**図1**のように「表形式」のデータを「集合指向」で操作するというモデルを前提にしています。普段Excelで事務をされている人ならそのイメージがあるので3日で使いこなせるようになるのに、多くの技術者は数ヵ月以上かかってもそこまで到達しない、と言ったら驚かれるでしょうが、それが実情なのです。10年以上SQLを扱っている技術者でも、LEFT JOINのバグを作ったうえで「バグであるということに気づいていない」ということもよく起こります(第16回[注5])。今までとは学ぶべきポイントが違うので、そこでいったんゼロから始めればいいのですが、頭のリセットがうまくいかずに「SQL嫌い」になってしまう人も少なくありません。

そうなると私の推奨と真逆の「できるだけSQLを使わずに処理しよう」という路線にいってしまいます。具体的には、単純なSQLのみ使う「ぐるぐる系」に走るか(第1〜3回)、あるいはO/RマッパをかましてプログラマがSQLを書かずに済ますという方向です(第15回[注6])。

O/Rマッパにまったく意味がないとは言いません。SQLの言語仕様には、たとえば小さく分割したコードを抽象化し、それを組み合わせて全体を構築する操作を視覚的にわかりやすく表現できないという欠点があります。そのため複雑なSQLになると、どこからどこまでをひとかたまりで読んだら良いのかを見分けるだけでたいへんになり、O/Rマッパがそれを解消してくれる面があるのは事実です。

しかし、SQLをプログラマから隠蔽(いんぺい)してしまうと実際にどのタイミングでどんなSQLが発行されるかがわかりにくくなり、プログラマが意図せず「ぐるぐる系」のSQLが出てしまうという、いわゆるN＋1問題などの弊害が起きやすくなります。RDBでの業務処理にはどうしても「複数のテーブルから関連するデータを集めて1つの処理をする」ようなものが多いのですが、O/Rマッパはこの種の仕事をするのに必要な複雑なSQLを書くのには向いていません。その結果、O/Rマッパに頼っていると「ぐるぐる系」の発想からなかなか脱却できないことになりがちです。

やはりSQLが扱う「表の集合操作のイメージ」

注5) 第16回：本誌2017年6月号。

注6) 第15回：本誌2017年5月号。

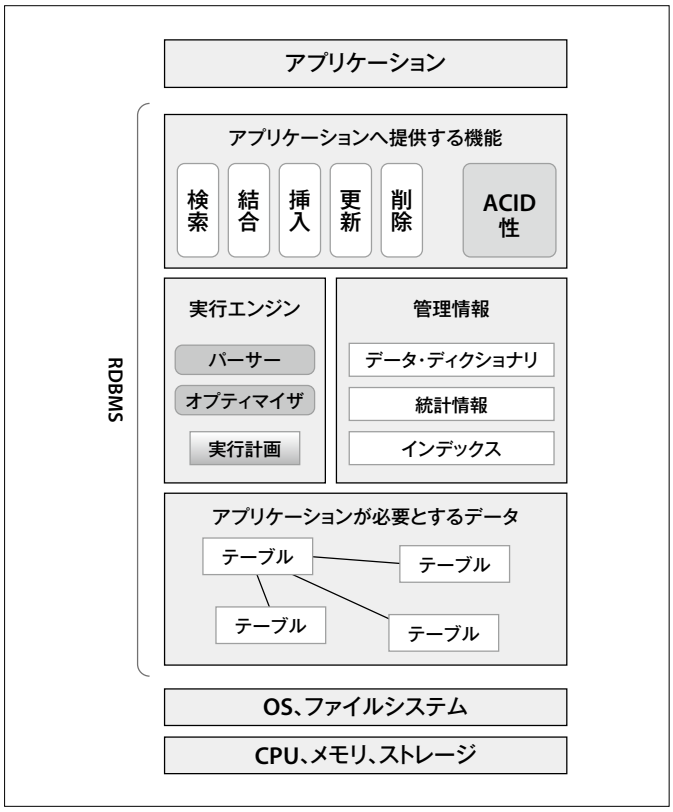
▼SQLを学ぶときはプログラミングの知識はいったん脳に置いておこう

を持ってSQLから逃げずに学んでいくべきです。そこで、「**ただしSQLに熟練するには専門的な勉強と経験が必要**」これがメッセージの3点目です。SQLそのものは簡単な言語なのですが、けっして何の努力もなしに活用できるわけではありません。

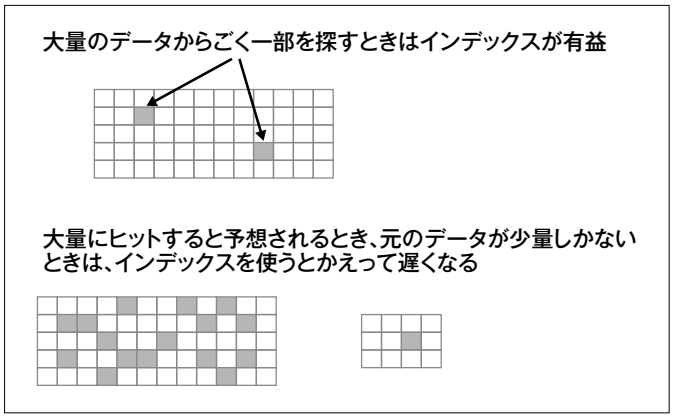
本格活用にはRDB内部のしくみへの理解も必要

COBOLでは言語仕様の中に簡単なインデックスによってレコード単位でファイルにアクセスするしくみがありましたが、それに比べるとRDBのしくみははるかに複雑です。**図3**がそのイメージで、アプリケーションが必要とするデータは相互に関連のあるテーブルとしてモデル化されています。それに対する検索／結合などの機能をアプリケーションに提供するにあたってACID性が保たれるようにロックやトランザクションの機能があり、それを効率よく実行するために統計情報やインデックスがあり、SQL文そのものはパーサー、オプティマイザを通して実行計画に変換して実行される、というこれらのしくみを理解していないと性能トラブルはなかなか解決できません。

▼図3　RDBMSは高度なデータ管理機能のカタマリ



▼図4　インデックスは常に役に立つわけではない



たとえばインデックスも常に役に立つわけではなく、**図4**のように役に立たない場合もあるため、現代のRDBは検索SQLを実行するときに統計情報をもとに「この検索ではインデックスが有益かどうか？」を判断してインデックスを使う／使わないを自動的に切り替えています。

「COBOLのファイルというのは、OSのファイルシステムを直接使っているようなものなんですか？」

「索引（インデックス）のしくみはあるからそこまで単純ではないけれど、RDBに比べると単純なのは確かだね。年配のITエンジニアには、そのCOBOLの感覚のままでRDBのことも大きなスト

レージぐらいにしか思っていない人もいるから困るんやけれど……」

COBOLの時代と違ってDBMS自体がインテリジェントに判断をして実行計画を組んでくれるのでプログラマにとっては楽な反面、性能トラブルを解決するときにはこれらの知識を持って実行計画を確認する必要があります。ときにはCPU、メモリ、ストレージの速度差というレベルまで考えることもあるぐらいで(第13回[注7])、これらを理解していないと、RDBを使っているのに「JOINは禁止」といった妙なルールを作ってしまうこともあります(第10、17回[注8])。SQLを本格活用するためにはRDB内部のしくみも含めて勉強していかなければなりません。問い合わせ言語であるSQLそのものは簡単な言語なのですが、それを活かして開発／運用コストダウンを実現するには、RDBを熟知したエンジニアを確保するという壁があるのです。

「そこなんですわ、ビジネスロジックもSQLで書くほうがえぇ、と言われて理屈としては理解できても、たとえばプロジェクトマネジャーや経営者のレベルでそれを聞いても自分がやれるわけじゃないからね……実際に現場で手足を動かすエンジニアにRDBを熟知した人間がおらんと、今までのやり方を変えるのはなかなか踏み切れへんでしょうね」と、五代さん。

「そんな人材をどう確保するかですが、内部で育成しようという意思はありますよね？」

「もちろんですとも！　だから生島さんにお願いしているわけで！」

担当を分けてAPIファースト・メソッドで開発を!

「RDBを熟知したエンジニアがいない」という壁を解消するためにお勧めしたいのが、「**DB担当とAP担当を分離したAPIファースト・メソッドでの開発体制**」をとること(第9回[注9])。これが4点目、最後のメッセージです。

ストアドプロシージャやファンクションによりDBからAP側に提供する「API」をまず作り、AP側は生のSQLを発行するのをやめて、そのAPIだけを使ってDBにアクセスします。ビジネスロジックの大半はDB側に移してAP担当はUIに専念し、DB担当は逆にDBに専念する形で担当範囲を切り分けるわけです。この場合のDB担当は最初からRDBについての十分な知識経験を持っていなくてもかまいません。大道君がそうであったように、「いつでも意見を聞ける、教えてもらえる相談役」が身近にいれば、いつもDBのことを考えているわけですから内部のしくみへの理解も短期間で深めていくことができます。

「実際、今はAPIファーストのDB担当としてやらせてもらってますけど、生島さんのおかげで1年前に比べるとすごく自信がつきました！」

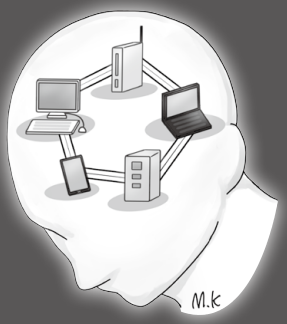
「私もそう思いますわ！」と五代さん。

さらにこの方式はアジャイル向きでもあり、スタブAPIをまず用意して仕様が変わりやすいUI部分を先行させ、UIが固まった時点でテーブル設計をしてスタブを実DBに切り替えることが可能です。当社では15年前から採用している方式で実績もあります。あなたの会社でも、APIファースト・メソッドを試してみませんか？

SD

▼「DBに専念＆相談役」で、若手を短期間でDBエンジニアに育成

注7) 第13回：本誌2017年3月号。
注8) 第10回：本誌2016年12月号、第17回：2017年7月号。
注9) 第9回：2016年11月号。



仮想化の知識、再点検しませんか?

使って考える仮想化技術

本連載は「仮想化を使う中で問題の解決を行いつつ、残される課題を整理する」ことをテーマに、小さな仮想化環境の構築・運用からはじめてそのしくみを学び、現実的なネットワーク環境への実践、そして問題点・課題を考えます。仮想化環境を扱うエンジニアに必要な知識を身につけてください。

Author
笠野 英松(Mat Kasano)
オフィス　ネットワーク・メンター
URL http://www.network-mentor.com/indexj.html

最終回　利用者リモート運用管理を適用した仮想環境

連載後半「仮想ネットワーク環境で使ってみよう～現実的な使い方」の9回目。連載の最終回です。

前回(本誌2017年7月号)は仮想環境の運用管理の具体的な実装例として、一般的な利用者管理APIのWebブラウザやセキュリティツールでWebサーバと仮想環境運用管理インフラを経由して、仮想マシンの利用部門(利用者)がリモート運用管理することを考えてみました。

今回は本連載のまとめとして、これまでの仮想マシン利用者によるリモート運用管理の実装例を使って、本格的な仮想環境を構築する場合にさらに必要な周辺機能、および、その具体的な仮想環境の応用事例について考えてみます。

本格的な仮想環境構築のための機能要件

大規模仮想環境では、相応の製品パッケージでリモート運用管理を含めたネットワーク管理を行うことになります。しかし、そうした製品パッケージはあまりにも機能がありすぎて、中小規模の仮想環境にとっては不要な機能が多く、使い勝手が悪くなります。

中小規模では最低限の運用管理機能があればよいのですが、本連載で解説してきた実装例は、あくまでも稼働する最低限の機能を解説したものです。運用管理のために最低限必要なトラブルシュートのためのツール、事前対策、事後対策、運用監視などについては言及していません。

たとえば、仮想環境のバックアップが一例です。これには第13回(本誌2017年6月号)で説明した、マイグレーションやスナップショットなどの仮想マシン障害対策機能、あるいは、仮想環境運用管理インフラのツールであるvirt-install/virsh/virt-managerを直接利用する、などでできます。運用監視には、一般のネットワーク監視ツール[注1]などを使うことも考えられます。

また、おおもとの仮想環境、具体的にはOSと仮想環境、を作ることも自動化すれば、さらに一般利用者のための仮想環境の利用・運用管理が効率的になります。

本連載で紹介してきた実装例はそうした「一般利用者が可能な限り運用管理に参画するための」実装の一部分ですが、この実装例をもとにすれば「完成形」を作ることが可能になります[注2]。

応用事例の形態と具体例

利用部門によるリモート運用管理を適用する、あるいは、それに適した仮想環境ネットワークの、一般的な利用形態や具体的な応用事例を考えてみましょう。

注1) (一般製品)PATROLCLARICE、PRTG
(フリー)MRTG、RRDtool、nagios、Hinemos、ZABBIX

注2) 参考:完全自動化仮想環境
http://www.network-mentor.com/vCenter.html

利用形態のタイプと必要条件

仮想マシンの一般のネットワーク利用は、通常の物理システムと同様に、マルチユーザモードでの利用です。ただし、この一般利用では、利用者を限定するかどうかで2つの利用が考えられます。その2つとは、パブリックサーバ利用と個別利用です(**図1**)。そのため、ファイアウォールでその制限フィルタ(利用者限定接続)を行わなければならないなど、追加設定が必要な場合もあります。

パブリックサーバ利用

仮想マシンを一般のクライアント－サーバ形式で接続を受け付けるための、ネットワークサーバとしての利用です。この場合には、すべての着信を許可します。

個別利用

仮想マシンをたとえば、BYOD[注3]用システムとして利用する場合で、1対1の個別制限接続となります。そのため、接続元のIPアドレスが対応する仮想マシンへ接続を許可する、preルーティング処理(本誌2017年7月号の第14回で解説)が必要となります。

利用形態のタイプと必要条件

本連載のような利用者部門によるリモート運用管理を適用した仮想環境マシン／ネットワークが実利用されている具体的な応用事例としては、以下に挙げるようなものがあります。

一般公開サーバとしての利用

Webサーバやメールサーバなどインターネット公開サーバとしての利用です。

一般の物理サーバと同じように、専用管理部門の管理となります。管理者のリモート運用管理接続は管理者限定セキュリティとなります。

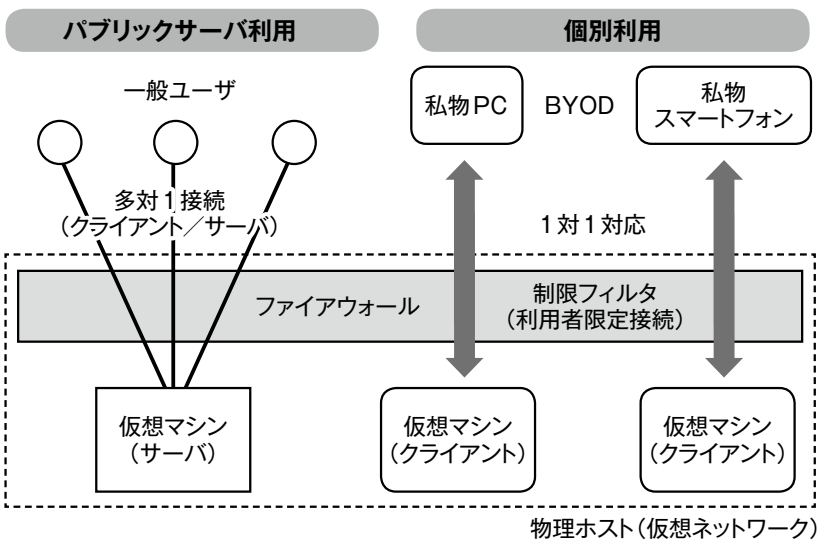
専用サーバとしての利用

部門・組織など、あるいは、データベースのサーバとして、特定の利用者、利用部門が専用利用する場合です(**図2**)。

組織外からの接続には制限が必要で、管理接

注3) Bring Your Own Device：企業のクライアント端末として私物PCやスマートフォン、タブレットなどを利用すること。

▼図1　仮想マシンの一般利用の形態



続は当該組織限定です。

✳一時的システムとしての利用

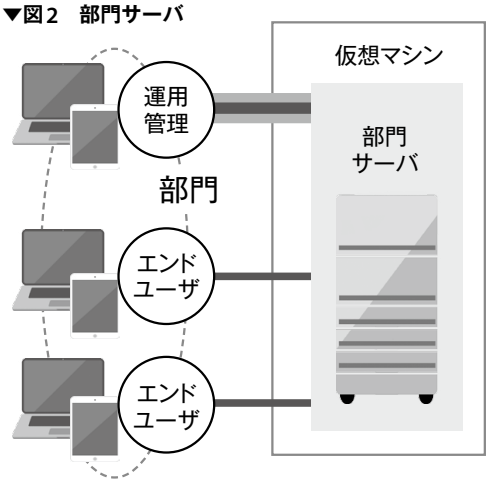
開発部門など、一定期間、特定のOSやアプリケーションなどを構築してソフトウェア開発やテストなどを行う場合です(**図3**)。また、教育・講習など、同様に一定期間、特定のOSやアプリケーションなどのインフラを構築して教育や講習を行う場合です(**図4**)。

いずれの場合も、その利用期間が終わればシステムを削除して消し去ります。なお、管理接続は当該部門の技術者に限定します。

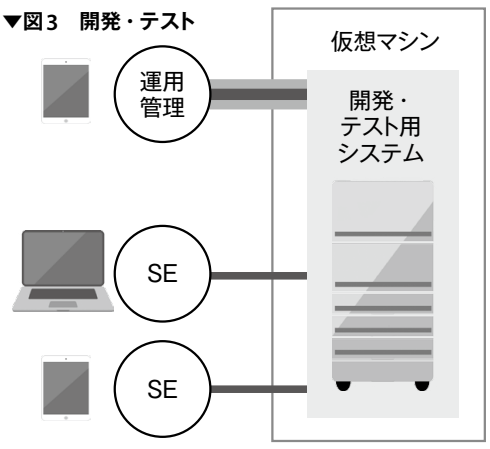
✳クライアントとしての利用

通常の社内クライアントシステムとして利用する場合ですが、BYODを利用した社員の社内システムとして利用することなども実際に行われています(**図5**)。この場合の管理・利用接続は利用者に限定します。

▼図2　部門サーバ



▼図3　開発・テスト



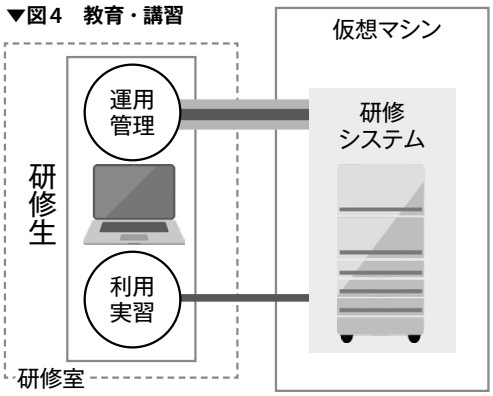
リモート運用管理のための特殊な接続

一般にリモート運用管理は、社内、サイト内に閉じた内部で行われることが多いのですが、社外からインターネット経由で仮想環境にアクセスすることも考えられます(**図6**)。

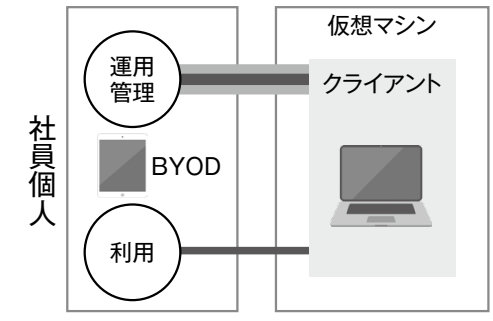
こうした場合、通常は、ルータ経由で直接物理ホストへアクセスすることになるかもしれません。このとき、セキュリティとしては発信IPアドレスやプロトコル(SSH/TCPやSSL/TCPやIPsecなど)、ユーザ／パスワード、双方向証明などがそのフィルタ要件となります。

セキュリティ強化のためには、さらに、コールバック／逆接続(物理ホスト側から管理端末

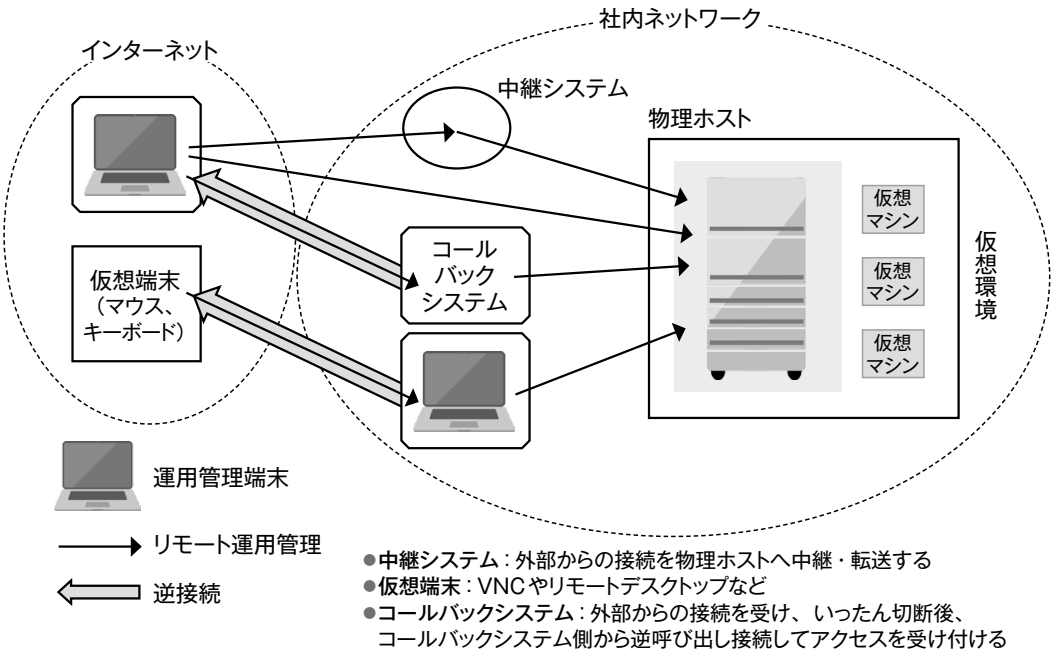
▼図4　教育・講習



▼図5　個人利用



▼図6　リモート運用管理のための特殊な接続



へ逆に接続する）などの接続形態に変更することも1つの方法です。この場合、物理ホストから直接逆接続するのではなく、社内のシステム経由でリモートの管理端末を中継接続するとよりセキュアです。管理端末を確実に特定できるからです。

あるいは、社内の別のシステム経由で間接的に物理ホストへアクセスする方法もあります。Hyper-VやESXiなどのように、管理端末が物理ホスト以外でなければならず、かつ、ルータ経由でのポート転送が複雑な場合です（Hyper-Vの場合、Hyper-Vマネージャー以外にもサードパーティ製の特殊な管理ソフトウェアもありますが）。

中継システムを使ういずれの場合も、リモートの管理端末から逆接続可能にするわけですが、その接続が切断された場合（二度と逆接続できなくなるので）、keep-alive（切断を感知するしくみ）を使って再接続するしくみが必要になります。そのシステムがWindowsであろうと、UNIX/Linuxであろうと、ユーザがログイン／ログオンせずにkeep-alive再接続できるようにサービス登録しておくことも重要なポイントです。

連載のおわりに

約1年半にわたって中小規模を対象とした仮想ネットワーク環境の利用・運用についてお話ししてきました。

大規模仮想環境ネットワークとは異なり、中小規模仮想環境ネットワークでは、さまざまな「リソース制限」が付きまといます。そのため、「リソースの効率的な利用」が必須です。大規模サイズ向けのベンダー製品はタスキに長し、フリーツールは帯に短し、であり、そのネットワークに適したカスタマイズをしにくい面があります。

本連載でお話した内容は、求められる全体システムの「フレームの節々」だけですが、骨格や肉付けなど「関連ツールや見栄え、利用しやすさ」を適宜「付け加えながら」、よりそのネットワークに適した「完成形」をつくることが可能です。

本連載はここで終わりますが、完成形に向けての骨格や肉付けの具体的な実装を、いろいろな場で紹介していきたいと思っています。SD

コミュニティメンバーが伝える Androidで広がるエンジニアの愉しみ

presented by
Japan Android Group
http://www.android-group.jp/

第17回 AIファーストでAndroidはどうなる? Google I/O現地レポート

Androidは世界で出荷される約9割のスマートフォンに搭載される標準OSです※。そのため、多くのAndroidアプリが開発され続けており、そして多くのエンジニアが活躍しています。Androidで広がる新しい技術に魅了されたエンジニアが集うコミュニティもあり、そこでは自分が愉しむための技術を見つけては発信しています。その技術の一幕をここで紹介します。

※Gartner Worldwide Smartphone Sales to End Users by Operating System in 3Q16

嶋 是一(しま よしかず)
NPO法人日本Androidの会 理事長

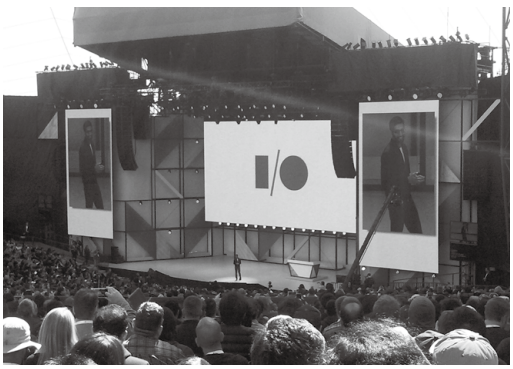
堀田 ほつた(ほった ほつた)
日本Androidの会 学生部

今年もGoogle I/Oが開催

Googleが毎年開催している「Google I/O」が今年も5月17～19日に開催されました。これは、Googleの技術に関する開発者のためのイベントです。米国Google本社近くのマウンテンビューに世界80ヵ国以上から7,000人を超えるGoogle技術者たちが集まり、熱狂の3日間を過ごしました。今回は本イベントに参加した筆者からの現地レポートをお届けします(**写真1**)。

Google I/Oでは、Androidに限らず、クラウドからAIまで、さまざまなGoogleの技術やプロダクトが発表されます。そこで発表したてほやほやの、まだ誰も試していない最新技術と最新情報を使うことができます。まだ誰も創り出したことがないプログラムをコーディングし、そして誰よりも先に動かすことができます。そこから発生する新しいイノベーションの夢を見つつ、集まった多くの開発者がチャレンジを始めます。これが熱気となります。しかし、事前情報がない技術は一筋縄では動きません。そのため、Google I/Oでは多くのGoogle社員(Googlerと呼びます)がいて、直接質問し、情報を聞く貴重な場も用意されています。今年のGoogleの技術動向を知るイベントと言っても過言ではないでしょう。

▼写真1 Google I/O Keynoteの様子



注目される発表とAI

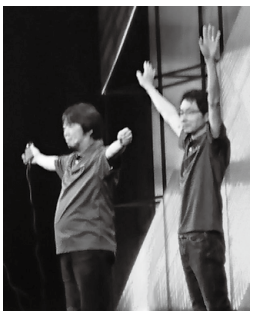
Android開発者目線で注目すべき発表は「KotlinのAndroid正式サポート」「TensorFlow Liteの登場」「TangoのVPS」「DayDream 2.0ユーフラテス」など多数ありました。しかし、イベントを通した最大のテーマは「AI」です。Googleが向かうのは「モバイルファーストからAIファーストへ」(今回のキャッチコピー)であり、膨大な利用者の情報をもとに、Googleのサービスやプロダクトに AIを活用して進化させるという強いメッセージでした。

AIに関しては日本に関連するネタがたくさんありました。1つは基調講演で話題になった「たこ焼き」です(**写真2**)。日本語がわからない旅行者が「たこ焼き」の看板を見つけたとき、Androidのカメラを通して画像検索することで、

▼写真2　たこ焼きとAI

▼写真3　Google Home（提供: Google）

▼写真4　ラジオ体操



どのような食べ物かを対話で表示してくれるというものです。AIの活用事例としての紹介でしたが、日本人の間では「たこ焼き6個130円」を見て「それは安すぎるだろう」という話題で盛り上がっていたのは余談です。

この技術はGoogle Assistantという Googleのサービスを用いており、Googleアプリ（旧Google Now、ホームキー長押しで起動するアプリ）を通して利用できます[注1]。このAssistantをGoogleアプリから音声入力で利用し、航空チケットを予約するような対話サービスや、音声入力からアプリケーションを起動させて利用するしくみを「Actions on Google」で開発できます[注2]。実際に対話はこのActions on Googleを通して、api.aiかActions SDKの対話機能を用いて構築します。ポイントはこのようなサービスを、我々でも開発可能であるという点です。なお、Google Assistant対応のデバイス「Google Home」が参加者全員に配布されました（**写真3**）。参加者への全プレ（おみやげ）もGoogle I/Oの特徴の1つです。

AIのもう1つの日本ネタは「ラジオ体操」です。「毎朝体操[注3]」というアプリは手に持ったスマートフォンの加速度センサーの値により、どれだけ上手に体操ができたかを採点するというものです。これは、事前に多くの人にラジオ体操を行ってもらい、その加速度センサーの値を大量に機械学習させ、できた学習モデルを用いてAndroid端末上でTensorFlowを用いて採点させるというもの。Google Japanの登壇者が海外の壇上でラジオ体操を踊りだすという前代未聞のショーにより、会場から拍手が湧き上がっていました（**写真4**）。

TensorFlow Lite

「TensorFlow」は、Googleが提供している機械学習のためのオープンソースのソフトウェアです。このツールは、PCやクラウド上に構築して利用するのが一般的です。今回Google I/Oで発表したのは「TensorFlow Lite」であり、これはAndroid端末やRaspberry Pi上でTensorFlowを動作させることができます。つまりネットワークと接続していなくても、TensorFlowで作成した学習モデルを端末上でオフラインのまま利用できるようになります。これにより、さまざまなアプリの中でAIの技術を活用することができるようになりました（まだ発表したてで、ほとんど誰も作っていませんが）。Android上には、このモデルが動作する

注1）利用可能な端末は一部です。最新端末でも順次アップデートが行われており、完了できしだい日本語の利用が可能となります。

注2）Actions on Googleはこちらから開発可能。
https://developers.google.com/actions/

注3）http://maiasa.jp/

ための「Android NN API（NNはニューラルネットワーク）」層が搭載される予定です（**図1**）。

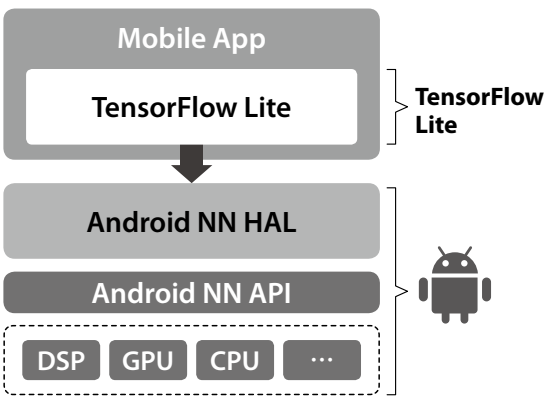
実は本誌2017年2月号の本連載「第13回 スマートフォンと人工知能がつながる未来」で、TensorFlowの端末実行について取り組んだ、古川新氏自身が寄稿した記事がありました。これと同等の取り組みがGoogleから公式な手法として発表されたことになります。メーカー含めて世の中的には、独自の取り組みがAndroid公式版としてリリースされることはよくあることです（Googleの上書きと呼ばれています）。

Kotlinの発表

Google I/Oの中で最もAndroid開発者を熱狂させたのが、モダンな開発言語である「Kotlin」のAndroid正式サポートでした（**写真4**）。

Androidのアプリケーション開発はJava SEで行います。しかし、Javaは世の中に出て約20年経ち、Android SDKが出て10年経っています。その間、開発しやすくコーディング量を抑える工夫がされた、モダンな言語が多数発表されています。とくにiOSについては、Objective-CからSwiftへの移行が3年前に発表されており、より新しい言語で開発できるようになっています。スマートフォンのアプリ開発者は、Android向けとiOS向けの双方を開発することがどうしても多くなります。そういう意味でも、

▼図1　TensorFlow Lite



▼写真4　Kotlin正式サポート発表の瞬間（両手を上げて拍手喝采。まさに熱狂の瞬間）

Androidは古きに取り残された状態だったのが、ここにきてKotlinのサポートに動いたことが、遅れを取り戻すものとして好意的に受け入れられました。

また、このKotlinの開発元のJetBrains社はAndroid Studio（AS）の開発をしており、AS 3.0でのKotlinのサポートと、すでに動作するAS 3.0 Canary版[注4]（味見版）をいち早く発表しています。

会場でのASデモストレーションで来場者を驚かせたのが、Java編集ウィンドウにあったJavaソースをコピーし、Kotlin編集ウィンドウへペースト操作すると、なんとJavaソースがKotlinに変換されてペーストされる、というものでした。それほどにKotlinとJava SEは好相性です。それもそのはず、KotlinもJava SEも実行するには「Javaバイトコード」に変換されます。Androidフレームワークから見れば、上でアプリがKotlinで書かれていようが、Javaで書かれていようが変わりがありません。そのため、安定して動作します。まだKotlinに関する日本語情報が少ない中、日本Kotlinユーザグループから「Kotlin助走読本[注5]」という読み物が日本語で公開されています。

セッション、サンドボックス、オフィスアワー、コードラボ

Google I/Oは、大きく4つから構成されてお

注4）https://androidstudio.googleblog.com/2017/05/android-studio-30-canary-2-is-now.html
注5）https://drive.google.com/file/d/0Bylpznm149-gTGRjOFRkWm9PODg/view

り、現地に行くべき理由を紹介します。

1つは講演形式となっている「セッション」です。155を超える技術セッションはすべて中継されており、YouTubeで閲覧できます[注6]。そのためセッションだけ聞くのならば、現地に行かなくても済みます。しかし現地に行くとわかるのが、セッションの人気具合や開発者の熱狂ぶりです。これらを共感すると、自分の「作るぞスイッチ」または「その技術を使ってみたいぞスイッチ」が入りまくります。

もう1つが「サンドボックス」です。これは技術のテーマを絞って展示を行っているドーム型のブースです。ここではセッションで紹介された最新技術を使った展示も多く、新技術が目の前で動いるのを体験すると、自分にも作れそうだと思ってしまうのがおもしろいところです。

そして「オフィスアワー」。これはテーマごとに100以上のブースが用意されており、そこでGooglerと1対1で質問や会話ができる場です。まさに現地でしか聞くことができない、その技術を創り出した人の思いや考え方を聞き出すことも可能です。

最後に、自分でコーディングの学習ができる「コードラボ」です。今年は85テーマにもおよび、中には完了するとRaspberry PiにAndroid Thingsが搭載されたキットがもらえるテーマもあり、学習意欲を掻き立てます。

Google I/Oに参加して

このイベントは、参加するための倍率の高さが有名です。今年は運良く筆者も当選したので現地の「熱狂」を体験できました。現地のGooglerと接することで、自分の開発意欲と技術興味が湧いてきます。この経験が何よりの糧となるため、やはり現地に行くことの大切さを知ることとなりました。SD

注6）https://events.google.com/io/

COLUMN

モバイルWeb技術「PWA」について

文●堀田ほつた

運良くGoogle I/Oアカデミック枠で当選し、英語嫌いで中学英語くらいしかわからず、海外に行くわけがないと思っていた大学生の自分が、日本Androidの会 学生部としてGoogle I/Oに初参加しました。

AndroidのみならずAMP（Accelerated Mobile Pages）やPWA（Progressive Web Apps）といったモバイルWeb技術の紹介がGoogle I/Oで積極的に行われていました。筆者はこれらに興味を持ち、現地セッションに参加しました。

AMPとは、Webページを高速に表示するオープンソースプロジェクトのことです。AMPページは平均1秒未満で表示され、データ量は10分の1となり、ユーザ体験が飛躍的に向上します。PWAとは、簡単に言えばモバイルWebをアプリ化する技術です。PWA対応Webページを表示し、ブラウザのメニューに出てくる「ホーム画面に追加」を行うと、まるでネイティブアプリのようにインストールできるのです。PWAの代表的な例は、2017年4月にリリースされたTwitter Liteです。Android版Twitterアプリが23MBのところ、Twitter Liteはわずか0.6MBで主要な機能が使えます。インドなどの新興国では、インターネット通信が給料と比べて割高で、データ通信量を抑えることが望まれています。また、低価格スマホは空き容量が少ないため、サイズが大きいアプリはインストールしにくいという事情があります。そういった国を中心にAMPやPWAを導入したサービスが爆発的に普及しています。Androidに限らず、AMPやPWAといった技術をぜひともみなさんにも知ってほしいと思います。

自分は現地に行ったものの英語力不足で、質問やほかの参加者とうまく交流ができなかったのが本当に悔しい限りでした。しかし、現地に行ったからこそ英語ができないことを痛感し、英語の勉強をしたい、開発をしたいという思いが心の中で高まってきました。Google I/Oでプレゼントされた GCP利用料700ドルクーポンと、共感したAMPやPWA技術を活用した世の中に役立つようなアプリを作ってみたいと思います。

日本Androidの会 学生部は、Androidに限らず開発意欲のある学生が集まるコミュニティです。Androidに縛られないコミュニティとして活動していますが、Androidにかかわる開発で活動することが多いです。直近では、学生限定で、メイドさんと学ぶ初心者Android勉強会を2017年7月29日（土）に開催する予定です。いつでも入部大歓迎です。Androidに限らず、開発意欲のある学生をお待ちしています!（http://student.android-group.jp/）

一歩進んだ使い方のためのイロハ

Vimの細道

第20回

mattn
twitter:@mattn_jp

外部コマンドを便利に使おう

Vimには、外部のコマンドと連携してさらに強力な機能を実現するしくみが備わっています。今回は標準入力の読み込み、標準出力への書き出し、フィルタ、ソートの方法、独自コマンドの作り方を解説し、より高度な編集術を身につけます。

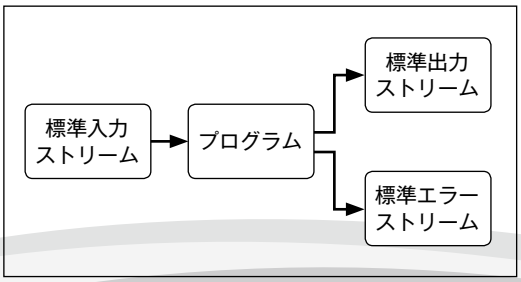
Vimは UNIX的

VimはUNIX向けのツールとうまく連携するテキストエディタです。:makeや:grepだけでなく外部コマンドをうまく使いこなすことができるようになれば、テキスト編集テクニックの幅が広がります。今回はVimから外部コマンドを便利に使う方法を紹介したいと思います。

まずは基本的なお話

プログラムは一般的に、標準入力ストリームと標準出力ストリーム、標準エラーストリームを扱えます(**図1**)。標準入力ストリームとは、次のように別のコマンドやファイル、端末(画面)といった物から出力された内容を読み込むための入力機能(ストリーム)を意味します。

▼図1　ストリーム




```
$ ls | my-command
```

またそのプログラムも別のコマンドやファイル、端末へストリームとして出力できます。

```
$ my-command | grep foo
```

この場合、my-commandから出力された標準出力ストリームがgrepコマンドの標準入力ストリームに流し込まれ、grepコマンドは標準入力ストリームからfooという文字列にマッチした行を見つけて標準出力ストリームである端末に書き出します。パイプでつなげられた各プログラムが入力を読みつつ出力するので、一連のコマンドの処理は並列に行われます。

Vimから外部コマンドを使う

外部コマンドを起動する

Vimから外部コマンドを起動するには、次のように!コマンドを使います。

```
:!my-command -f config.json
```

たとえば、プロジェクトフォルダに置かれたテストデータ更新用スクリプトを実行するといった場合に便利です。

コマンド部分に%を使うことで、外部コマン

ドに編集中のファイル名を引き渡すことができます。たとえば編集中のファイル名のバックアップを作るのであれば、次のように実行します。

```
:!cp % %.bak
```

編集を始めてからまだ:wを実行していないのであれば、ファイルには変更を行う前の内容が保存されています。Vimで変更を始めてしまったけれど、保存する前に最後の状態をバックアップしておきたいといった場合に便利です。

gitコマンドを使うこともできます。筆者はVimからgit連携を行うプラグインを使用していません。Vimからコミットを行う場合は次のように実行しています。

```
:!git commit -a -m "update README.md"
```

!コマンドを実行すると、Vimは起動したコマンドの終了を待ちます。プログラムを止めたい場合はCtrl-cをタイプします。Windowsに限ってプログラムの終了を待たない実行方法が用意されており、:!startで起動できます。

```
:!start notepad %
```

外部コマンドの標準出力を読み込む

外部コマンドの標準出力を読み込むには:r!コマンドを使います。たとえば、カレントディレクトリのファイル一覧を編集中のバッファに読み込む場合は次を実行します。

```
:r!ls
```

:r!コマンドでは、現在いる行の次の行からテキストが足されます。行の途中でコマンドの出力を取り込みたい場合には適しません。コマンドの処理結果を現在の行の末尾に付け足したい場合には、次のように実行します。

```
:r!date<NL>-join
```

<NL>はCtrl-v Ctrl-jをタイプして挿入します(^@と表示されます)。なおdateコマンドは、Windowsでは少し違う動作をします。書式は異なりますが、次で同様の結果を得られます。

```
:r!date /T<NL>-join
```

コマンドの標準入力に書き込む

標準入力を読み込むコマンドに編集中のテキストを書き込むには、:w !コマンドを使います。

```
:w_!my-command
```

注意しないといけないのは、:wと!の間にスペース(_)が必要ということです。スペースを入れずに:w!my-commandと実行すると、my-commandというファイルができてしまいます。

筆者がよく使うのは、Vimでメールの本文を書きながら外部コマンドで送信する方法です。

```
From: me@example.com
To: you@example.com

Do you love Vim?
```

メールの本文を記入したら次のように実行します。

```
:w !sendmail -t
```

Windowsの場合はmsmtpというコマンドを使うことができます。メールのFrom/Toから送信元や宛先が判定されるので、よく確認してから実行してください。

フィルタ

編集中のバッファ内容を外部コマンドの標準入力に書き出し、外部コマンドから出力されたテキストでバッファ全体を置き換えます。たとえば編集中のファイルに行番号を付けたいのであれば、行番号を付与するコマンドnlを使って次を実行します。

```
:%!nl
```

また、ビジュアル選択した部分だけにフィルタを実行することもできます。この場合は行単位の選択(ラインワイズ選択)でなければなりま

せん。後述のためにも説明すると、ビジュアル選択モードには3つの選択方法があります(**表1**)。フィルタで実行可能なのはこのうちラインワイズだけになります。ほかの選択モードで実行しても開始行と終了行を使ったラインワイズと同じ結果となり、選択部分が外部コマンドの出力結果で置き換えられます。

ソート

フィルタの応用例です。テキストの中に書かれた行をソートするには次を実行します。

```
:%!sort
```

数値で始まる行を並べ替えるのであれば-nオプションを付与します。ソートしたあとに重複行を除去するのであれば、外部コマンドのuniqを使います。

```
:%!sort | uniq
```

ただし最近のVimであれば、内部コマンドに:sortが用意されています。あまり知られていませんが、実はこの:sortはすごい機能をたくさん持っています。:sortコマンドのオプションは**表2**のとおりです。範囲が指定されない場合はファイル全体をソートします。

また:sortコマンドはパターンを取ることができます。パターンにマッチした部分以降で並べ替えが行われます。

```
ABBA
Duran Duran
The Beatles
The Rolling Stones
Styx
```

このテキストで:sort /^\(A |The \)*/を実行すると、先頭のAやTheが抜かれた部分だけでソートされます。

▼表1　ビジュアル選択モードの種類

名前	説明	開始キー
キャラクタワイズ	文字選択	v
ラインワイズ	行選択	V
ブロックワイズ	矩形選択	Ctrl -v

```
ABBA
The Beatles
Duran Duran
The Rolling Stones
Styx
```

また逆に、先頭がAやTheで始まる行を並び替えるには、:sort /^\(A \|The \).*$/ rのように、行をマッチさせるパターンを書いてrフラグを付けます。この際マッチしなかった行は、並べ替えられない状態で先頭に移動します。:sortはかなり強力なコマンドですので、ぜひマスタしてみてください。

キャラクタワイズでフィルタを使いたい

先に説明したとおり、キャラクタワイズ選択ではフィルタが期待したとおりに動作しません。しかしながらそうしたい場合もありますので、その際のテクニックを紹介します。

キャラクタワイズ選択にて、選択されているテキストをいったんヤンク(コピー)してしまい、そのテキストを外部コマンドに渡し、その結果で選択していた部分を置き換えるということを行います。

たとえば選択したテキストを大阪弁に変換したいとします。大阪弁への変換は、筆者作のosaka[注1]を使います。インストール方法などはREADME.mdを参照してください。現在入力されているテキストを次のとおりとします。

```
今日は金曜日です。残業かもしれません。少し遅れます。
```

注1) URL https://github.com/mattn/osaka

▼表2　内部コマンド:sortの機能

コマンド	オプション
:sort!	逆順で並べ替え
:sort i	大文字小文字を無視
:sort n	行が数字で始まる場合は数字として並べ替え
:sort f	行が浮動小数点であるとして並べ替え
:sort x	16進数として並べ替え(0xはなくても良い)
:sort o	8進数として並べ替え
:sort b	2進数として並べ替え
:sort u	並べ替え後uniq(重複除去)を実行

この「残業かもしれません。少し遅れます。」をキャラクタワイズ選択しているとします。外部コマンドにテキストを渡すために一度ヤンク(y)してしまいます。ヤンクすると無名レジスタ"に格納されるので、置換を行うためにgvで前回の選択状態に戻します。さらにcをタイプして変更モードに移行し、続いてCtrl-r =をタイプして式挿入を行います。ここで先ほど得たレジスタの値を使います。Vim scriptに外部コマンドを実行する関数systemが用意されていますので、次のようにレジスタの内容を使って実行します。

```
=system('osaka', @")
```

これを実行するとテキストが次のように変換されるはずです。

```
今日は金曜日です。残業かもしれまへん。ちびっと遅れまんねん。
```

手数が多いのでマップにしてみます(**リスト1**)。テキストを選択し,osakaをタイプすると大阪弁に変換されます。別の例ですと、

```
今日は 21 June 2017 です。
```

▼リスト1　標準語を関西弁に変換するマップ

```
vnoremap ,osaka ygvc<c-r>=system('osaka', @")<cr><esc>
```

▼リスト2　日付の書式を変換するマップ

```
vnoremap <leader>D ygvc<c-r>=substitute(
 system('date --date=' . shellescape(@")
 . ' +"%Y/%m/%d"'),"¥n","","g")<cr><esc>
```

と書かれたテキストの「21 June 2017」を「2017/06/21」に変換するには、**リスト2**のようにUNIXのdateコマンドを使ってマップを作ります。dateコマンドは必ず改行コードを出力するので、substitute関数で最後の改行を取り去っています。このコマンドには注意点があります。無名レジスタ"を使うのでヤンクしていたはずのテキストが書き換わってしまいます。回避方法についてはまた別の号で紹介したいと思います。この手法を使えば、外部コマンドを使っていろいろなテキスト変換が行えるようになります。

◆ ◆ ◆

今回はVimから外部コマンドを使う場合の基礎と、外部コマンドを使った独自フィルタを作ってみました。外部コマンドを使うことで、Vimだけでは実現できない高度な編集を行えます。ぜひ独自のコマンドを作ってみてください。SD

Vim自報

権限のないファイルを更新

/etcの下にある設定ファイル等を調査していると、ちょっと中身を見るだけのためにVimを起動してみたものの、ガッツリと編集してしまい、あとから権限のないファイルであったことに気付いて:wで書くことができず、致し方なく権限のあるフォルダに一時的に保存して凌いだ、なんて経験のある方もいるかと思います。

そんな場合も、UNIXのコマンドを組み合わせることで解決できます。

```
:w !sudo tee %
```

Vimは%を現在のファイル名(例：/etc/hosts)に置き換え、バッファの内容をsudo tee /etc/hostsの標準入力ストリームに書き出します。sudoコマンドは権限を昇格したあとに、teeコマンドを実行して標準入力ストリームを読み、与えられたファイル名に書き出します。これを簡単に実行できるsudo.vimというプラグインも存在しますが、今後のためにもこの方法を覚えておくと良いでしょう。

書いて覚える Swift 入門

第28回 WWDC 2017特集

Author 小飼弾（こがい だん）　twitter @dankogai

新ハードウェア紹介に終始したキーノート

今回は久しぶりに新ハードウェアがどっさりと発表されました。それも2時間30分超えのキーノートの大半の時間を使って。新OSもつつがなく発表されたとはいえ、iOS 11はとにかくmacOSはHigh Sierraとは、あまり新しそうに見えません。

とは言っても現行MacはCPUがKaby Lakeにアップデートされただけのように見えますし、iMac ProもHomePodも発売開始は今年末であり、その意味で真の新製品はiPad Pro 10.5ということになるでしょう。筆者も1台入手しました(**写真1**)。

なぜAppleはこれらの新製品発表の場としてSpecial EventではなくWWDCを選んだのでしょう？——ほかの誰よりも開発者達がそれを欲していた、というのが筆者の読みです。とくにiPad Pro 10.5は、よりMac寄りに近づくことが確定的になったiOS11 for iPadのレファレンス・マシンとしての役割を担っています。ドラッグ&ドロップ可能なマルチタスク画面にDockまで。iPadはコンテンツを消費するデバイスからコンテンツを制作するデバイスへという流れをより鮮明化させてきました。これでXcode for iPadがあれば、Macフリーなエコシステムができあがるようにすら見えるほど。

▼写真1　真の新製品iPad Pro 10.5

本番は「プラットフォーム一般教書演説」

むしろWWDCとしての本番は、Platforms State of the Union[注1]でした。ベタな直訳で「プラットフォーム一般教書演説」。機械学習APIであるMLKitと拡張現実APIであるARKitは他社の後追いのようにも感じられますが、本連載で何度も繰り返しているように、Appleは最も保守的なプラットフォーム企業。そのAppleが標準APIを用意するというのは、それが今後の業界の標準となることを示唆せずには入られません。

注1）URL https://developer.apple.com/videos/play/wwdc2017/102/

Swift Playgrounds meets robots

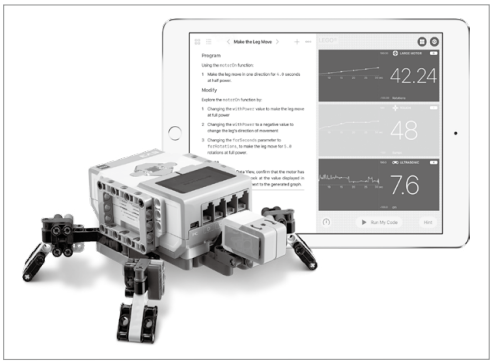
本連載の主題はAPIではなくSwiftという言語。そこに焦点をあてるとSwift Playgroundsでドローンや LEGO MINDSTORM EV3をプログラムできるようになるという発表(**写真2**)は、IT業界よりも教育業界にとって朗報でしょう。筆者の次女が通っている中高一貫校の高等部ではiPadが必須の教材となっているのですが、彼女たちもきっとその恩恵を受けることでしょう。

Swiftで書き直された Xcode Source Editor

とはいえ本連載をお読みの読者のほとんどは、XcodeでSwiftしているはずです。そのXcodeは9でどう変わったか。一番の注目は、Swiftで全面的に書き直されたソースエディタでしょう。妙な言い方になりますが、ソースコードを「理解」するようになっています。言葉で言うより、スクリーンショットをご覧いただいたほうが早いでしょう(**図1、図2**)。

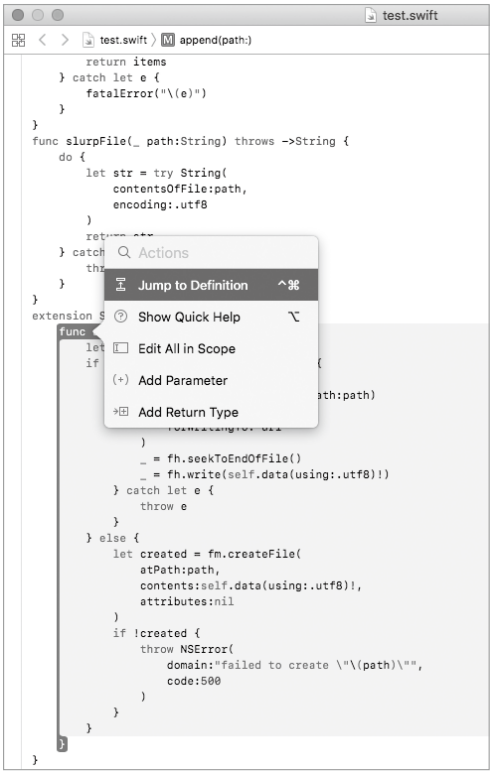
iOSアプリでもあるSwift Playgroundsと同様、ソースの構造を適切に解釈したうえで、その構造に即したコンテキストメニューが表示さ

▼写真2　LEGO MINDSTORM EV3をiPadでプログラミング



▼図1　コンテキストメニュー表示(その1)

▼図2　コンテキストメニュー表示(その2)



れます。

圧巻なのは、プロトコルとの組み合わせ。本連載でもSwiftはProtocol Oriented Languageであると繰り返し申し上げてきましたが、言語自身はとにかくこれまでのエディタのサポートは貧弱なもので、不適合なら“Type T does not conform to protocol 'P'”というエラーメッセージを表示する程度でしかありませんでした。

それが、**図3**のとおりに変わります。

プロトコルを追加した時点で適合性がチェックされるところまでは今までどおりですが、足りないプロパティをすべて洗い出したうえでスタブを追加してくれるのです。あとはそれを埋めるだけ。ああ、PONS[注2]を開発中にこの機能があったらどんなに楽だったか！

エディタの機能が増えると心配なのが「それでますますもたつくのではないか」。うれしいことに、機能が豊富になったにもかかわらず、編集からビルドまでほぼすべての面で高速化しています。とくに長いコードが入ったPlaygroundは、Xcode 8までは明らかにもたついていたのが、ほぼ何をしても瞬殺という感じで非常に快適。正直、High Sierraを待たずに先に正式版リリースしてほしいぐらい。

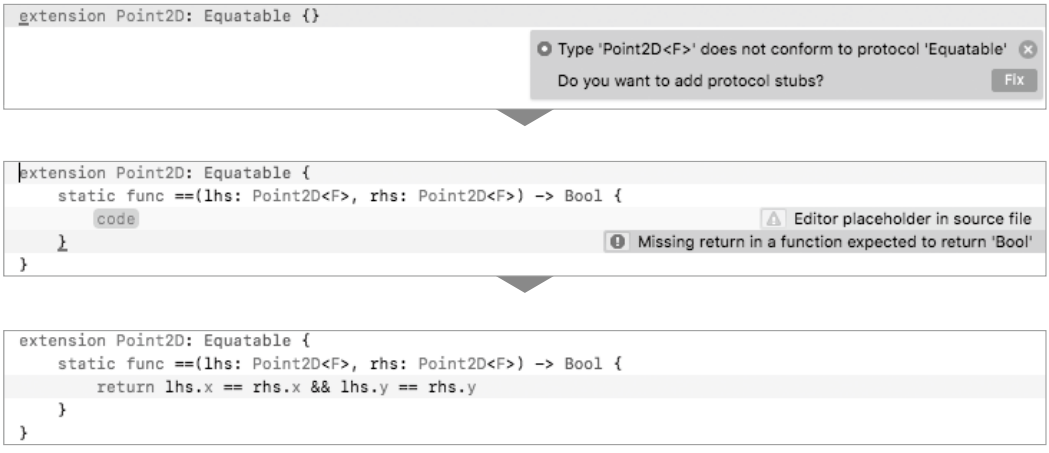
Swift 4「簡単なことはより簡単に」

開発用のMacもXcodeも明らかに改善されたのを確認できた今回のWWDCですが、肝心のSwift 4はどうでしょう？「一般教書演説」でも少し紹介されていましたが、やはりここはWhat's New in Swift[注3]でもう少し詳しく見ていきましょう。

言語仕様自体は、Swift 3からそれほど変わっていません。Swift 3のリリースのときと同様Swift 4のリリースでは同時にSwift 3.2がサポートされます。ここまでは同様ですが、前回はXcodeにコンパイラーを2種類搭載してそれを実現していたのに対し、今回は同じコンパイラーでそれを実現していることからも2→3ほどの変化ではないことがうかがわれます。とはいうものの、3.xのxの増分では済まない違いも確かに存在します。

- private宣言がより自然になったので、fileprivateはほぼ不要に。プレゼンテーションでは「これでアクセスコントロールに関してはおしまい」という言い方でこの変更についての説明を締めくくっていた

▼図3　エラーメッセージの変化



注2）URL https://github.com/dankogai/swift2-pons
注3）URL https://developer.apple.com/videos/play/wwdc2017/402/

・プロトコル宣言で新たなプロトコルを宣言しなくても、プロトコルを合成できるように。P & Qと書けば、「プロトコルPとプロトコルQ双方に適合するタイプ」と解釈される。あとのコードサンプルにも実例が出てくる

・Sequenceの要素を指定するのに、Iterator.Elementと書かずにElementだけでよくなった

たとえばSwift 3では、

```
extension Sequence
  where Iterator.Element: Equatable
{
    func containsOnly(
        _ value: Iterator.Element
    ) -> Bool {
        return !contains { $0 != value }
    }
}
```

と書かなければならなかったのが、Swift 4では、

```
extension Sequence
    where Element: Equatable
{
    func containsOnly(
        _ value: Element
        ) -> Bool {
        return !contains { $0 != value }
    }
}
```

で済むように。Protocol Oriented Programmingがますますはかどります。内部的には、

```
protocol Sequence {
  associatedtype Iterator: IteratorProtocol
  // ...
}
protocol IteratorProtocol {
  associatedtype Element
  // ...
}
```

となっていた定義を、

```
protocol Sequence {
  associatedtype Element
  associatedtype Iterator: IteratorProtocol
    where Iterator.Element == Element
  // ...
}
protocol IteratorProtocol {
  associatedtype Element
  // ...
}
```

としたとのことです。見ればなるほどですね。

StringがCharacterのCollectionに

しかし一番歓迎な変更は、StringがCharacterをElementとするCollectionになったことでしょう。今まで、

```
let animals = "🐶🐮🐷💎"
for c in animals.characters {
    print(c)
}
```

として.charactersプロパティにアクセスしないと取り出せなかったのが、Swift 4以降は、

```
let animals = "🐶🐮🐷💎"
for c in animals {
    print(c)
}
```

と書けるということです。もちろん今までどおり、バイト単位でバラす.utf8や、Unicode単位でバラす.unicodeScalarsは健在です。

ここで「Unicode単位」と言いましたが、Swift 4におけるCharacterは合成文字もきちんと1文字として扱います。なので、

```
let flags = "🏴‍☠️🏳️‍🌈"
flags.count                // 2
flags.unicodeScalars.count // 4
flags.utf8.count           // 16
```

となります。なお、今までどおり.charactersも用意されているので、Swift 3以前のコードもそのまま動くはずです。

あと、Pythonと同様の"""で複数行Stringリテラルもサポートされます。次に紹介するCodableと組み合わせると、文字列にとどまらず入り組んだデータの初期化もはかどること請け合いです。

Codableプロトコル

筆者が最も感銘を受けたのが、Codableプロトコルの追加です。本連載の第25回で、JSONタイプを作成しましたが、今後こういったタイプはほぼ不要になるのです。実例を見てみましょう。

```
struct Point2D
  <F:FloatingPoint>
{
    let x:F
    let y:F
}
```

という簡単なタイプがあったとします。これをJSONで扱いたいとしたらどうするか？　こうするだけでよいのです。

```
struct Point2D
  <F:FloatingPoint & Codable> : Codable
{
    let x:F
    let y:F
}
```

そう。ストアドプロパティがすべてCodableでありさえすれば、そう宣言するだけであとは何もしなくてよいのです。では実際に使ってみましょう。まずはPoint2DからJSONデータへ。

```
let p = Point2D(x:42.0, y:0.195)
let j = try JSONEncoder().encode(p) // Data型
String(data: j, encoding: .utf8)
// {"x":42,"y":0.19500000000000001}
```

次はその逆。

```
let pp = try JSONDecoder()
  .decode(Point2D<Double>.self, from:j)
```

ここでおもしろいのは、.decodeの第一引数が型になっていることです。tryしているので、JSONがこの型に適合していなければthrowすることは言うまでもありません。

鋭い読者であればすでに察しているとは思いますが、当然[Codable]も[String:Codable]もCodableなので、文字列や数値はすぐにこの恩恵を受けられますし、Swift 4以降は標準搭載のJSONEncoderのみならずMessagePackEncoderやYAMLEncoderがリリースされることも期待されます。

ほかにもRangeがstart...endと書かなくても、start...だけでstartから最後の要素までの意味になったりと、Swift PlaygroundsとあわせてますますカジュアルにSwiftを使えるようになるのは確実そうです。

APFS Update

Leopardに Snow Leopard、Lionに Mountain Lion。そしてSierraにHigh Sierra。AppleがOS Xもとい macOSのバージョンをそう命名するときには、外見はあまり変わらずとも内部がガバッと変わるものと相場が決まっていますが、今回の内部は格別に大きい。ついにAPFSがmacOSにも正式導入されるのです。ただし、macOS SierraやiOS 10.3のAPFSそのままではありません。What's new in Apple File System[注4]を見てみることにしましょう。

一番変わったのは、Unicode Normalizationの有無。High SierraのAPFSはNormalizationを行います。本連載第26回のときの検証コードをHigh Sierra BetaのAPFSで使ってみたところ、こうなりました。

注4）URL https://developer.apple.com/videos/play/wwdc2017/715/

・Perl版：POSIX API

```
食べないでくださーい

食べないでくださーい
食べないよー

かばん.txt:\x{304b}\x{3070}\x{3093}.txt
```

・Swift版：Foundation

```
食べないでくださーい

食べないでくださーい
食べないよー

かばん.txt: ["\u{304B}", "\u{306F}", "\
u{3099}", "\u{3093}", ".", "t", "x", "t"]
```

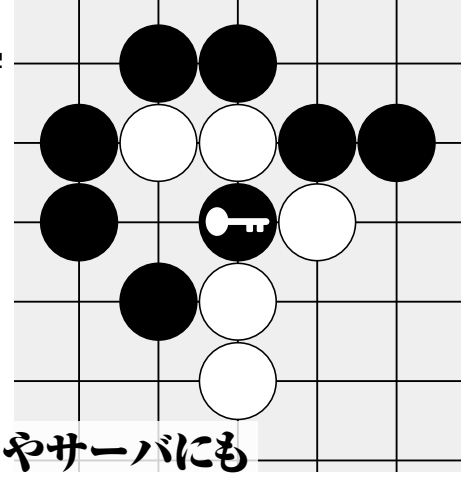
やや奇妙なことに、Perlを含めPOSIX API経由ではNFCが、Foundation経由ではNFDがそれぞれファイル名として使われていますが、`\u{304b}\u{3070}\u{3093}.txt` と `"\u{304B}\u{306F}\u{3099}\u{3093}".txt` が同じファイル名だと認識されています。

次回予告

というわけでSwiftを取り巻く環境の今後の方向性も見えたところで今月の記事は結びです。次回はSwift 4を見据えつつ、Swift 3でももちろん有効なSwiftyな記事をお届けする予定です。SD

セキュリティ実践の基本定石

すずきひろのぶ
suzuki.hironobu@gmail.com



みんなでもう一度見つめなおそう

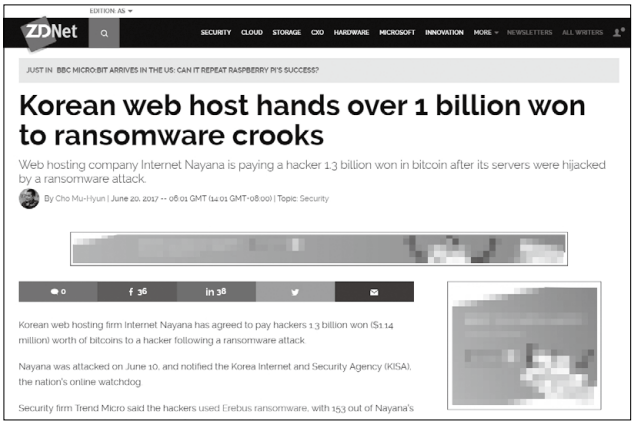
【第四六回】ランサムウェアの脅威はLinuxやサーバにも

ランサムウェア「WannaCry」が世界的に猛威をふるい、国内外のメディアにも大々的に取り上げられました。前回の本連載でも問題の本質を考えるという内容で取り上げました。それらはPC環境で感染が広がった事例であったため、「ランサムウェアはPC固有の問題だ」「ほかのコンピュータ環境での感染は理論上可能でも現実的ではない」と思われていないかと、筆者は懸念しています。

ランサムウェアの身代金、13億ウォン支払いへ

韓国のホスティング企業NAYANA社がランサムウェアに感染し、13億ウォン（約1億2,700万円）の身代金を払うかもしれないという記事を見つけました（**図1**）。本原稿執筆時では、すでに13億ウォンの身代金を払い復旧に向かっています。

◆**図1　「韓国のWebホスト企業が10億ウォン以上をランサムウェア犯人に支払う」との報道**[注1]



人質を取って身代金を要求するという犯罪を考えると、ランサムウェア「WannaCry」のように、相手にかかわらず低料金で定額要求するというケースは、極めて例外的です。犯罪といえども経済合理性は働くわけですから、被害者の支払い能力によって最大限の要求をするはずです。一律定額を要求するWannaCryのほうが極めて珍しいケースと言えます。別な言い方をすると、WannaCryの犯人は身代金を要求する犯罪とはどういう犯罪なのか、ということをまったく考えていないで実行に及んでいたと言えるでしょう。

NAYANA社はホスティングしていた顧客のデータを人質にとられ、犯人と巨額の身代金の交渉をせざるを得ない状態になってしまいました。ホスティングはLinux系のサーバで行っており、被害にあったのは153サーバ、3,400のビジネスのWebサイトとデータとのことです[注2]。

NAYANA社が2017年6月14日に出したステートメントが同社サイトに掲載されていました[注3]ので、そこから6月22日時点で

注1）"Korean web host hands over 1 billion won to ransomware crooks"
http://www.zdnet.com/article/korean-web-host-hands-over-1-billion-won-to-ransomware-crooks/
「ランサムウェア被害で13億ウォン支払いへ、韓国ウェブホスティング企業」（2017年6月21日）
https://japan.cnet.com/article/35103036/

注2）"Web Hosting Company Pays $1 Million to Ransomware Hackers to Get Files Back"
http://thehackernews.com/2017/06/web-hosting-ransomware.html

注3）ランサムウェア被害にあったNAYANA社のステートメント　**http://notice.nayana.com/**
NAYANA社カスタマーセンターに掲載されている社長からの経過説明メッセージ
http://www.nayana.com/bbs/set_view.php?b_name=notice&w_no=961

発表されていることの要点を抜き出してみます。

- 6月10日の午前1時に攻撃および被害が発覚した
- 「Erebus」というランサムウェアが使われた
- すでにKISA[注4]などには届け、調査などの支援を受けた
- 当初の要求は50億ウォン（約4億8,800万円）相当のビットコインでの支払いであった
- 交渉の末、18億ウォン（約1億7,500万円）まで切り下げた
- 同社の現金資産は4億ウォン（約3,900万円）であり、最初の交渉で支払い可能金額として4億ウォンを提示した
- 最終的には会社売却などを含め13億ウォン（約1億2,700万円）で交渉がまとまった
- 支払いに関しては3回に分けた身代金の支払いのつど、復号鍵を受け取ることにした
- 3回の取引が終わり、すべての鍵は入手できた
- 現在復旧中で最大10日かかる見込み

あらゆる方面への可能性を模索していたようですが、犯人と金額を交渉し、身代金を支払って復号鍵を手に入れる方法しかなかったようです。

犯人は巨額の身代金を要求し、また、被害企業と複数回に渡る金額交渉をしています。「ランサム（身代金）」を要求する犯罪としてみた場合、標準的とも言えるプロセスになっていると感じます。また、身代金交渉の際の最初の金額のふっかけ方、あるいは金額の落としどころなど、交渉スタイルを見ても、犯人はターゲットを絞ったうえで周到に計画したのではないかと感じます。

もちろん、犯人は交渉のときに自分をかくすため、Tor（匿名通信システム）などを使っているのは想像がつきます。短い時間で交渉を進め、最後は13億ウォンをビットコインで入手したわけですから、犯人側はかなりのものです。

筆者はインターネット上にある資料を調べていく間、クライムスリラー映画のあらすじを誰かがネット上に書いたのではないかと思うぐらいでした。

あと、この企業の社長が交渉時に相手（犯人）に送ったメッセージが掲載されていたのですが、「20年間かけて築き上げた会社だが、たとえデータが戻っても会社は運転資金もなく、信用も失い、さらに訴訟対応に追われるので会社が破産するのは回避できないだろう」と書いてあった部分は読んでいてたいへん心が痛みました。

ランサムウェアの復旧プロセス

NAYANA社は顧客向けに、どのようなプロセスを経て復旧させるのかの説明も書いていました。

(1) 侵害サーバのデータをWindows Server 2012で動作するコンピュータにコピーしたあと、復号プログラムによるリカバリを実行（2～5日）
(2) リカバリ完了後、データをバックアップ。実サービスが可能なようにパーミッションの調整などの環境設定を行う（1～2日）
(3) 二次感染を予防するためのリカバリデータの整合性チェックを実行（1～2日）

括弧の中の日数は作業に必要な時間です。ここからわかるように、残念ながらこの会社ではそれまでランサムウェア対策としてのバックアップなどは作成していなかったようです。この会社は適切なバックアップが行われていなかったためランサムウェアだけではなく、火災などで物理的にサイトが破壊された場合も同様な問題が発生したことになります。

データをリカバリさせるプラットフォームにWindows Server 2012を選択しているのは、万が一のとき、Linuxのバイナリやコマンドなどが動作しないように配慮しているのだと思われます。

Linuxのランサムウェア

これまでもLinuxのランサムウェアがないわけではありませんでした。もちろんその犯罪の内容もほかとなんら変わりません。レベルもいろいろです。これもPCと同様です。

注4）"KISA（Korea Internet & Security Agency）とは、韓国の政府外郭団体で、同国での情報セキュリティを担当している組織。
http://www.kisa.or.kr/eng/main.jsp

たとえば、「Linux.Encoder.1」というランサムウェアは、作成者が暗号の知識に乏しかったため、正しいセッション鍵と初期化ベクトルIVを生成できず、復号できてしまうというバグを持っていました。数学的な意味で言っている乱数と、乱雑さの量であるエントロピーの違いを理解していないプログラマが作りがちな致命的なバグでもあります。

中には最初からファイルをディレクトリごと消しておいて、戻してほしかったら金を振りこめと脅すランサムウェアもありました[注5]。もちろんお金を払ったところで、すでに`rm -fr`をしているのでデータは戻りません。

世界で最大級の情報セキュリティ関連のカンファレンス「RSA Conference 2017」で、マイクロトレンドの研究者が“Not UNIX to Windows Anymore: A First in a Booming Ransomware Industry”という内容[注6]でUNIX系（LinuxやOS X）でのマルウェアについての議論をしています。また、同じ研究者が“Unix: A Game Changer in the Ransomware Landscape?”という記事[注7]も書いています。

筆者は、これまでLinux、とくにサーバ向けのランサムウェアが少なかったのは、技術的な問題でもなんでもなくランサムウェアのプラットフォームとして、金をかせぐ魅力に欠けていたからだと考えています。これまでPCで儲けていたレベルの人間がチャレンジしていなかった、つまり単純に効率の問題だったのではないかと考えています。

ランサムウェアErebus

今回使われたErebusに関してはマイクロトレンド社がすでに分析しています[注8]。

もともとErebusファミリーは2016年9月に現れたマルウェアで、最初に現れたときはWindows XP、Windows Vista、Windows 7をターゲットにしていました。次は2017年2月に現れ、今度は同じベースでUNIX向けになって現れたとのことです。

NAYANA社では、Linux kernel 2.6.24.2という古いカーネルでシステムを動かしており、DIRTY COWの脆弱性があったのではないかと指摘されています（DIRTY COWは2016年12月のアップデートですでに対処されています）。

DIRTY COWは、「LinuxのELF形式の実行ファイルは実行時メモリにマッピングされるが、実行途中にファイルを変更すると、その変更した内容が実行されてしまう」という脆弱性です。たとえば、一般ユーザが書き込み可能な実行ファイルをrootで動かしているとします。その実行最中に（一般ユーザが）実行ファイルを書き換えると、マッピングされているメモリ中の実行コードが変化することにより、その結果、root権限で任意のコードが動いてしまうということになります。

また同社は、Apache 1.3.36[注9]、PHP 5.1.4という2006年にコンパイルされたものを継続して使っていたため、脆弱性を抱えたまま運用していたということになります。これらの古いバージョンの脆弱性に対するツール類も出回っており、それらを利用すれば外部から侵入は可能であろうと思います。

このような理由から、外部からマルウェアが侵入し、root権限を奪われ、Erebusにより暗号化されたというストーリーは十分に理解できるでしょう。

これでLinuxもターゲットに

今回米ドルにして百万ドルを越える身代金が支払

注5）“Hacked Redis Servers being used to install the Fairware Ransomware Attack”
https://www.bleepingcomputer.com/news/security/hacked-redis-servers-being-used-to-install-the-fairware-ransomware-attack/
上記サイトには、すべて消去するひどいスクリプトが掲載されています。

注6）“Not UNIX to Windows Anymore: A First in a Booming Ransomware Industry”
http://blog.trendmicro.com/go-beyond-next-gen-xgen-rsa-2017/

注7）“Unix: A Game Changer in the Ransomware Landscape? (February 13, 2017)”
http://blog.trendmicro.com/trendlabs-security-intelligence/unix-a-game-changer-in-the-ransomware-landscape/

注8）Erebus Resurfaces as Linux Ransomware
http://blog.trendmicro.com/trendlabs-security-intelligence/erebus-resurfaces-as-linux-ransomware/

注9）Apache 1.3.36の脆弱性一覧
https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-66/version_id-51592/Apache-Http-Server-1.3.36.html

われたという事実が、世界中に告知されてしまいました。これは犯罪を行う側にモチベーションを与えてしまったということです。

Webサーバの書き換えが発生するのは日常茶飯事です。これがもしWebサーバ上で動作しているWebアプリケーションのコードを変更できるような状態であれば、そこを突破口に外部から任意のコマンドを実行させる確率は極めて高いと言えます。実際にはWebアプリを稼動させていなくても、たとえば、デフォルトでPHPが実行できるようにセットアップされていれば、外部からバックドアのプログラムを植え付けられる可能性はあります。

本誌2017年5月号では、Apache Struts 2の任意のコードを実行できる脆弱性[注10]について取り上げました。もし今後、このような脆弱性が現れれば、情報漏洩やサイト書き換えという形ではなく、ランサムウェアが埋め込まれ、サイト内の顧客情報や商品情報などが暗号化されて身代金を要求されるケースも出てくるということを示唆しています。

セキュリティアップデートとバックアップが定石

脆弱性を突かれないようにセキュリティアップデートをするのは必須ですが、Apache Struts 2のように、ほぼゼロデイ攻撃のようなケースも考えられます。最悪なケースを考えてバックアップを取る必要があります（バックアップ後は隔離されていなければいけません）。

バックアップを取るのは手間もコストもかかりますが、バックアップがなかったために犯人に多額の身代金を払うことになった状況を見ていると、「バックアップのコストのほうがはるかに安かったろうに」と思います。言うは易し行うは難しと言いますが、会社存亡のレベルまでになってしまった今回の事件を見る限り、「定石をきちんと押さえておかなければ」と強く感じざるを得ませんでした。SD

注10）JVNDB-2017-001621　Apache Struts2に任意のコードが実行可能な脆弱性
http://jvndb.jvn.jp/ja/contents/2017/JVNDB-2017-001621.html

SOURCES

レッドハット系ソフトウェア最新解説

第11回

Cockpitで始めるLinuxサーバ管理

RHEL7にはLinuxサーバの管理を簡単に始められるCockpitが含まれています。今回はCockpitの概要と利用方法を紹介します。

Author 小島 啓史(こじまひろふみ)
mail：hkojima@redhat.com
レッドハット㈱ テクニカルセールス本部 ソリューションアーキテクト

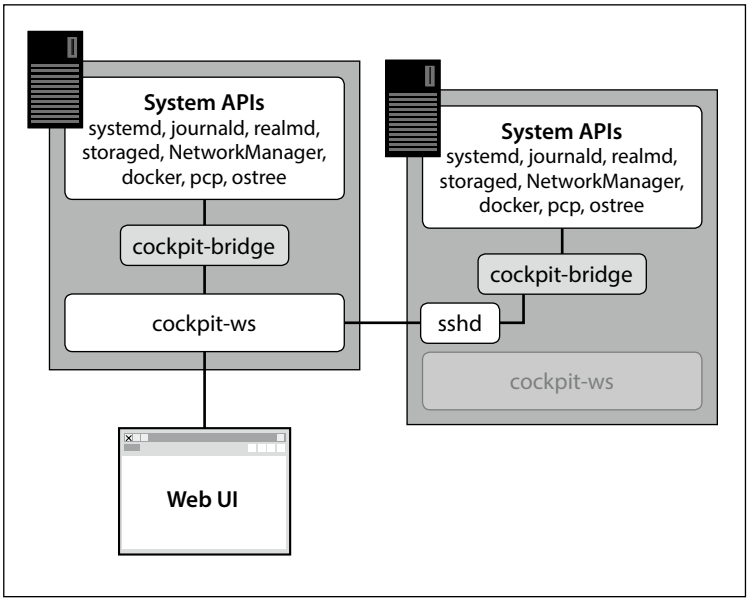
Cockpitの概要

Cockpit[注1]は、Linuxサーバの対話型管理インターフェースです。「Linuxの専門家でなくてもサーバを管理できるようにする」ことをコンセプトに開発されており[注2]、Cockpitは可能な限りLinuxのデフォルトインストール状態で動作するようになっています。そのため、サーバ管理者はCockpit関連ソフトウェアをインストールして起動するだけで、Cockpitを利用した管理業務(コンテナ起動／ストレージ管理／ネットワーク構成／ログ検査など)を始められます。

Cockpitはリモートサーバを含めた複数台のサーバを、1つのWeb UIから管理できます。**図1**は2台のサーバをCockpitにより管理しているイメージです。この図では2台のサーバそれぞれにCockpit関連のソフトウェアがインストールされています。そのうち1台のサーバのcockpit-wsサービス(Web UIを提供するサービス)を利用して、ローカル／リモートサーバを管理するようにしています。Cockpitの仕様では、cockpit-bridgeというユーザプロセスを経由してサーバの各種情報取得や設定変更を行うサービスを呼び出しています。また、リモートサー

注1) URL http://cockpit-project.org/
注2) URL http://thewalter.net/stef/misc/cockpit-devconf-2014-talk.pdf

▼図1 Cockpitによる複数サーバの管理イメージ



▼図2　目的のリポジトリの有効化

```
# subscription-manager register --auto-attach
# subscription-manager repos --disable="*"
# subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-extras-rpms
```

バへの接続にはSSHを利用しています。CockpitによるSSH接続にはパスワード認証のほかにも、サーバであらかじめ設定されている公開鍵認証やシングルサインオン認証を利用できます。**図1**はかなり簡略化したイメージとなりますので、より詳細なイメージを確認したい場合は公式リポジトリの画像[注3]をご参照ください。

Cockpitのインストールと起動

Cockpitはいろいろな種類のLinuxにインストールできます[注4]が、今回はRHEL7を利用してCockpitをインストールしていきます。あらかじめ用意した1台のRHEL7サーバをサブスクリプション管理サービスに登録し、Cockpitをインストールするために必要なリポジトリだけを有効化します(**図2**)。

続いてCockpit関連ソフトウェアをインストールし、Systemdを用いてcockpit.socketの有効化・起動を実施します(**図3**)。Cockpitはフットプリントをできる限り少なくするように開発されており[注5]、socketタイプのUnitを使うことでオンデマンドのサービス起動を行うようにしています。最後にFirewalldを利用している場合は、firewall-cmdコマンドでCockpit関連のポートを空けておきます。Cockpitはデフォルトで TCPの9090番ポートを使用しますが、必要に応じてポート番号の変更や特定のサーバからのアクセスしか受け付けないように設定[注6]もできます。

▼図3　Cockpit関連ソフトウェアのインストールと起動

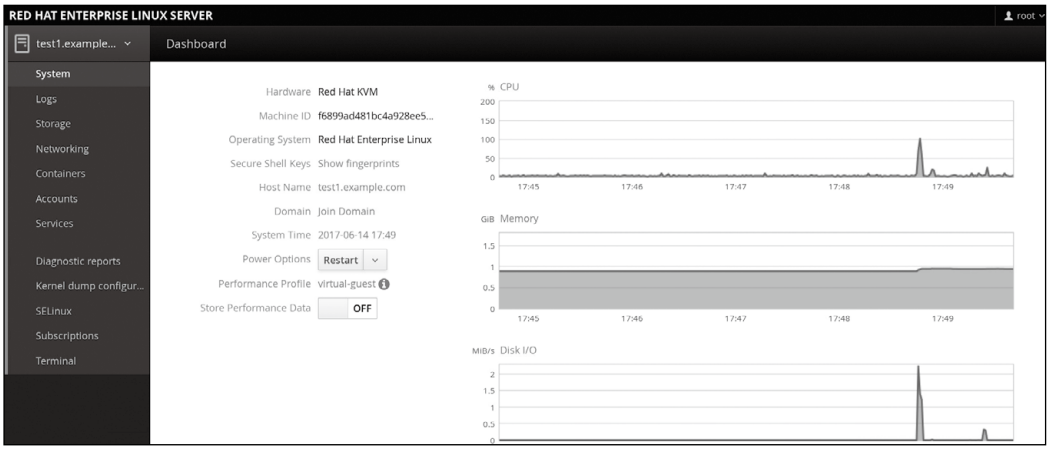
```
# yum -y install cockpit*
# systemctl enable --now cockpit.socket
# firewall-cmd --add-service=cockpit
# firewall-cmd --add-service=cockpit --permanent
```

Cockpitの利用方法

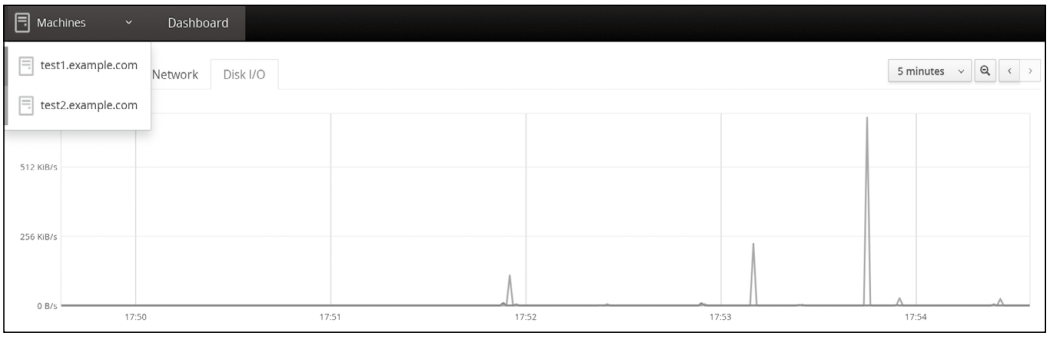
Cockpitを起動すると、Firefox/Google Chrome/Internet ExplorerなどのWebブラウザでCockpitにアクセスできるようになります。https://ip-address-of-server:9090にアクセスするとCockpitのログイン画面が表示されますので、ローカルアカウントまたはシングルサインオン認証[注7]を利用してログインします。ログインすると**図4**のようなホーム画面が表示されます。ホーム画面からシステム概要を確認できるほかに、ログ／ストレージ／ネットワーク／Dockerコンテナ／アカウント／システムサービス／kdump／SELinux／サブスクリプションの情報確認や設定変更、sosreport[注8]の取得ができるようになっています。設定変更などに伴い権限昇格が必要な場合は、あらかじめsudoやPolicy Kitの設定をしておく必要があります[注9]。また、「Terminal」からWebブラウザ上で端末を開けるので、直接コマンドを実行することもできます。なお、RHEL7標準のCockpitではKubernetesの管理を行うことができませんのでご注意ください。その場合には、OpenShiftを利用してcockpit-kubernetesをインストール[注10]する必要があります。

注3）URL https://github.com/cockpit-project/cockpit/blob/master/doc/cockpit-transport.png
注4）URL http://cockpit-project.org/running.html
注5）URL http://cockpit-project.org/ideals.html
注6）URL http://cockpit-project.org/guide/latest/listen.html
注7）URL http://cockpit-project.org/guide/latest/sso.html
注8）URL https://access.redhat.com/ja/solutions/78443
注9）URL http://cockpit-project.org/guide/latest/privileges.html
注10）URL https://access.redhat.com/solutions/2206041

▼図4　Cockpitのホーム画面



▼図5　2台のサーバのリソース利用率を表示するダッシュボード



Cockpitではリアルタイムのパフォーマンスデータ（CPU、Memory、Disk I/O、Network Traffic）をグラフ表示で確認するとともに、Performance Co-Pilot（PCP）[注11]を利用してデータのアーカイブ保存[注12]ができるようになっています。ホーム画面の「Store Performance Data」をONにすると、サーバ上でPCPのアーカイブ機能を持つpmloggerサービスが自動的に起動されます。

Cockpitのダッシュボードでは、各サーバのリソース利用率をグラフで比較できます。**図5**ではtest{1,2}.example.comという名前を持つ2台のサーバのリソース利用率をグラフ表示しています。リモートサーバを追加する場合は、前述したようにSSHの認証情報を利用します。追加した後は、ホーム画面からリモートサーバの各種情報の確認・変更ができるようにもなります。

Cockpitで表示されるこれらの画面を、ほかのWebアプリケーションの表示画面に統合[注13]することもできます。その場合には、**リスト1**のようにHTMLからCockpitの各コンポーネントのURLを呼び出します。**リスト1**ではローカルホストの例を表示していますが、Cockpitに登録してあるリモートサーバの情報を表示したい場合は「@localhost」を「@FQDN-of-remote_server」などに変更します。

注11）URL http://red.ht/2tolOZr
注12）URL http://cockpit-project.org/guide/latest/feature-pcp.html
注13）URL http://cockpit-project.org/guide/latest/embedding.html

▼リスト1　Webアプリケーションからの Cockpit 呼び出し例

```
<!-- ローカルホストのシステム情報 -->
<iframe src="https://localhost:9090/cockpit+app/@localhost/shell/index.html"/>

<!-- ローカルホストのログ情報 -->
<iframe src="https://localhost:9090/cockpit+app/@localhost/system/logs.html"/>

<!-- ローカルホストのサービス情報 -->
<iframe src="https://localhost:9090/cockpit+app/@localhost/system/services.html"/>

<!-- ローカルホストのコンテナ情報 -->
<iframe src="http://localhost:9090/cockpit/@localhost/docker/console.html"/>
```

▼図6　カスタムパッケージの追加例

```
$ wget http://cockpit-project.org/files/pinger.tgz -O - | tar -xzf -
$ cd pinger
$ mkdir -p ~/.local/share/cockpit
$ ln -snf $PWD ~/.local/share/cockpit/pinger
$ cockpit-bridge --packages
 ……(中略)……
pinger: /home/.../.local/share/cockpit/pinger
 ……(中略)……
```

Cockpitのパッケージ

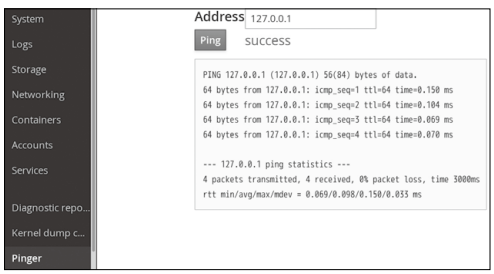
Cockpitはカスタムパッケージにより機能を拡張できます。Cockpitの標準機能は標準パッケージ[注14](前述したcockpit-kubernetesもパッケージの1つ)により提供されており、「cockpit-bridge --packages」コマンドを実行すると現在どんなパッケージがあるか確認できます。cockpit-bridgeはこうしたパッケージの確認のほかにも、CockpitのWeb UIから各種システムサービスへの中継にも利用されます。

非管理者がCockpitにカスタムパッケージを追加する場合は、ホームディレクトリに「.local/share/cockpit」というPATHのディレクトリを作成して、カスタム機能を提供するHTMLファイル(オプションでJavaScriptファイル)と、バージョン／参照するHTMLファイル名／UIでの表示名などを記載したManifestファイルを置く必要があります。こうした手順に沿って、CockpitのWeb UIに特定ホストにpingを打つ機能を持つカスタムパッケージを追加してみます(図6)。

カスタムパッケージの追加後にCockpitに再ログインすると、「Pinger」という名前のメニューが追加されており、ping機能を利用できることを確認できます(図7)。この例ではCockpit標準のAPIをHTMLからcockpit.js[注15]経由で利用し、pingプロセスを実行するための関数を呼び出しています。メニューを削除する場合は、「~/.local/share/cockpit」にあるpinger(この例ではシンボリックリンク)を削除してCockpitに再ログインします。

このようにCockpitは簡単に利用できることに加えて、機能のカスタマイズにも対応しているためLinuxサーバの管理を始めるには手頃なツールと言えます。興味がありましたら、ぜひとも試してみてください。SD

▼図7　CockpitのWeb UIでのping実行



注14) URL http://cockpit-project.org/guide/latest/packages.html

注15) URL http://cockpit-project.org/guide/latest/cockpit-spawn.html

UnityからGNOMEへの移行状況

Ubuntu Japanese Team／株式会社 創夢
柴田 充也（しばた みつや）

先々月号や先月号の本連載でも紹介したように、Ubuntu 17.10以降はグラフィカルシェルをUnityからGNOME Shellへと変更することになりました。今回はその進捗や実作業の流れを紹介します。

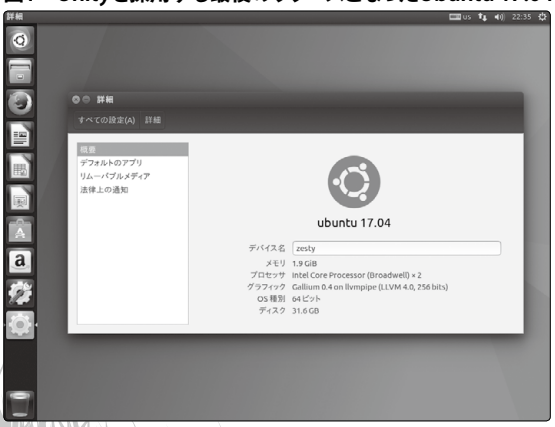
GNOME Shellを採用するということ

*「Canonicalは*Unity*をやめるぞ！」*

この突然のアナウンスに4月のUbuntuコミュニティはドッタンバッタン大騒ぎでした。しかしながらアナウンスの時点では「2017年10月リリースのUbuntu 17.10から変更される」「Ubuntu GNOMEの成果を流用する」以上の情報はありませんでした。そのあと、4月後半から17.10の開発が本格的に開始し、UnityからGNOME Shellへの移行にあたっての懸念点・手順などさまざまな課題を洗い出し議論するフェーズが開始しました。6月までには問題が整理され、リリースに向けての道筋が見えてきましたので、今回はその流れを紹介しましょう。

まずは状況について簡単に整理しておきます。最初のポイントは**「UbuntuはGNOMEベースのLinuxディストリビューション」**であることです。確かにFedoraなど、ほかのGNOMEベースのLinuxディストリビューションとは見た目や操作性が大きく異なります(**図1**)。しかしながら、デスクトップ環境としてのGNOMEが提供する各種ソフトフェア群やGNOMEプロジェクトがかかわっているインフラストラクチャー・ライブラリを利用してデスクトップを構築しているという点において、ほかのディストリビューションとの違いはあまりありません。

図1　Unityを採用する最後のリリースとなったUbuntu 17.04



ほかのディストリビューションとの差異の最たるものが**「GUIシェルとしてGNOME Shellではなく Unityを採用している」**ことです。GUIシェル（グラフィカルシェル）とはGUI環境においてユーザインターフェースを提供するソフトウェアを意味します。ユーザの操作に合わせてメニューを表示し、アプリケーションを立ち上げ、ウィンドウを配置するといったデスクトップに表示するもろもろの処理を担うものが、広い意味でのGUIシェルです。ディスプレイマネージャーからログインすることで開始するGUIセッションにおいて、どのGUIシェルを使うかを決定します。ただしUnityもGNOME Shellもデスクトップの表示のすべてを担っているわけではありません。どちらも事実上は、ほぼ「アプリケーションランチャー」に近い位置づけとなっています。

Ubuntuは、リリース当初からGNOMEベースでした。GNOMEのリリース周期に合わせて、Ubuntuのリリースが行われていましたし、当初のデスク

トップシェルはGNOMEそのものでした。その後、いくつかの技術的・経済的・政治的な理由により、Ubuntu独自のグラフィカルシェルであるUnityが開発され、GNOME Shellではなく Unityが採用されるにいたっています[注1]。

Unityのインターフェースを実現するにあたって、GNOME関連の下回りにもかなり変更を加えてきました。ソフトウェアによってはUbuntu用にフォークしたうえで実装しています。GNOME Shellに戻るためには、これらの「Ubuntu独自パッチ」をどのように扱うかが問題となるでしょう。幸いUbuntuのフレーバーには、GNOME Shellを採用した「Ubuntu GNOME」が存在します(**図2**)。つまりUbuntuの公式リポジトリに存在するパッケージだけで、GNOME Shellを採用したディストリビューションを作った実績はあるということです。あとは要・不要なパッケージのピックアップ、「見た目」をどこまでUnityに合わせるかの判断、最初からインストールするソフトウェアの取捨選択などは最低でも考える必要があります。これらを半年後の17.10のリリースまでに行わなければなりません。たとえ間に合わなかったとしても、遅くとも次のLTSのリリースである2018年4月までには解決する必要があるわけです。

17.10の開発が開始されて最初に行われた4月18日のミーティングでは、次の内容が決定されました。

- Ubuntu独自パッチの扱いはLPもしくはIRCで個別に精査する
- PPAをベースにGNOME Shellへの切り替えの調査を行う
- 切り替えはAlpha 2を目標とする

数年に渡って実装されたUnityのアレコレを、半年で整理する戦いが始まりました。

標準パッケージを見なおそう

Ubuntuのリポジトリは、そのライセンスやCanonicalによるサポートに合わせて「main・restricted・universe・multiverse」の4つのコンポーネントに分かれています。Ubuntuインストール時に最初からインストールされるパッケージ、つまりインストーラのISOイメージに同梱されるパッケージは、「main・restricted」に所属していることが求められます。つまりGNOME Shellへと移行するためには、GNOME Shellやその周辺パッケージをuniverseコンポーネントからmainコンポーネントへと移動する必要があるのです。

単に場所を移動するわけではありません。Ubuntuのサポート期間を通して、不具合や脆弱性の対応ができるかどうかを判断する必要があるのです。既知の脆弱性は存在するのか、既存の不具合はどれくらい残っているのか、ソフトウェアの開発コミュニティは活発か、パッケージのビルドテストには通っているか、パッケージが十分にメンテナンスされているか、そのパッケージを導入することでほかのコンポーネントもmainに必要になるかなど、いくつもの項目をパスして初めてmainへの移行が承認されます。

GNOME Shellの場合は、最低でもGNOME Shell本体とシステム設定ツール、ソフトウェアキーボードにそれらが依存しているライブラリパッケージの移動が必要でした。とくにシステム設定ツールについては、歴史的経緯からUnityではGNOMEのそれをフォークして使っていました。今回ようやくGNOMEと同じ「システム設定」を使うことになります(**図3**)。

6月中旬の時点で、上記はすべてmainリポジトリ

図2　17.10はこのUbuntu GNOME 17.04の成果をベースに開発される

注1) Unityの歴史については、本連載の先月号(2017年7月号)が参考になります。

へと移行しています。現在は、gnome-tweak-toolsなどのツールやgnome-mapsなどのアプリについても、最初からインストールすべきかどうかの議論が始まっています。

Waylandはどうするんだい?

Fedoraが標準でWaylandを採用しているように、UbuntuでもWayland対応が課題となっています。

Ubuntuを含む、ほぼすべてのメジャーなLinuxディストリビューションは、ディスプレイサーバとしてX Window Systemを採用してきました。30年以上使われ続けてきたこのXは、さまざまな機能拡張を経て、現在では巨大で複雑なソフトウェアとなっています。そのためXに代わるよりコンパクトなディスプレイサーバを求めて、Waylandが開発されたのです。Waylandそのものは、ディスプレイサーバとアプリケーションの間の描画に関する通信プロトコルであると同時に、コンポジッタやクライアントで使用するライブラリの実装でもあります。GNOME ShellにおけるWaylandへの対応とは、Xの代わりにディスプレイサーバとなるライブラリをリンクしたコンポジッタを利用することです[注2]。

GNOME Shellはすでに対応を完了していたため、あとはディスプレイマネージャー側の対応だけという状態でした。Unity8対応の影響でLightDMからWaylandセッションを起動できない状態になっていましたが、17.10開発の早い段階で修正が行われています(**図4**)。今後はテストインフラを整えて、Wayland上の不具合を洗い出していくことになります。すでにいくつかのアプリがWayland上では動作しないことがわかっているものの、順調に不具合の対応が進めば、17.10の時点でWaylandが標準となるかもしれません。Wayland移行における現在の最大の問題点は、リモートデスクトップ関連が利用できなくなるということでしょう。

注2) Waylandと並ぶ次世代ディスプレイサーバにMirがあります。けれどもこれは別の物語、いつかまた、別のときに話すことにしましょう。

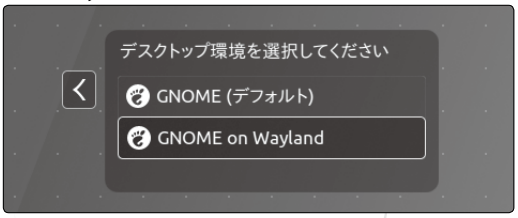
図3 17.10のGNOMEコントロールセンターの見た目は、これまでのシステム設定とほぼ同じ

LightDM vs. GDM

GUIセッションを管理するディスプレイマネージャー(DM)として、Ubuntuやそのフレーバーはこれまでを使ってきました。従来のDMと比べてLightDMは、ログイン画面などのUI部分を独自に拡張可能なGreeterとして分離している点が特徴です。これにより個々のデスクトップ環境は、Greeterのみを開発するだけで同じDMを共有できるのです。XubuntuやLubuntuは同じLightDMを使いつつ、GTK+ Greeterとテーマ機能によってそれぞれ独自のログイン画面を構築しています。GNOMEは以前より独自のDMであるGDMを使っていました。よってGNOMEへの移行にあたっては、LightDMを使い続けるかGDMに移行するかという議論が発生するわけです。

LightDMはGDMよりも作りがシンプルです。またGDMに比べるとテストコードが充実しており、品質管理もそれなりに行われています。他のフレーバーとの整合性を考えると、Greeterやテーマで切り

図4 17.10では通常のGNOMEセッションとWaylandセッションを選択できるようになる



分けられるLightDMを使い続けたほうが有利です。またゲストセッションの機能はLightDMにしかありません。つまりUbuntuとして考えた場合、LightDMに軍配があがります。

しかしながら、GNOME Shellのロック画面はGDMに依存しているという問題がありました。さらに調査を続けるとLightDMでGNOME Shellのロック画面対応を行うには、思ったよりも時間がかかることも判明してきました。それらの状況を踏まえて、6月上旬の時点で17.10ではGDMを採用しつつ、ほかのフレーバー向けにLightDMもメンテナンスし続けることを決定しています。ゲストセッション機能のGDMへの移植は、17.10に間に合うかどうか未定です。

来るGNOME 3.26と去るGTK+2

GNOMEは3月と9月の半年ごとのリリースを行っています。偶数のバージョン番号がリリース版で、奇数が開発版であるため、リリース番号は1つ飛びとなります。2017年4月時点での最新版が3.24であり、9月に3.26がリリースされる予定です。そこで問題になってくるのが、17.10では3.26を採用するかどうかという点です。17.04では多くのコンポーネントが最新の3.24を採用していましたので、3.26に上げていくこと自体はそこまで手間ではありません。とくに3.26はHiDPI対応の拡充や描画の高速化など、利便性の高い更新が行われる予定です。

ただし近年のGNOMEはリリース直前に大きな変更を行うことがあるため、以降の判断は9月まで先送りとなりました。3.26ではいくつかのソフトウェアのビルドシステムがautotoolsからmesonへと移行する見込みです。さらに、そろそろ将来的なGTK+2パッケージの削除を考える時期になってきています。

日本語入力はどうしよう

日本のユーザにとって最も気になる問題が日本語入力関連でしょう。ここ数年のUbuntuの日本語環境はFcitxとmozcを採用していました。しかしながらGNOMEはIBusにべったりな実装となっています。実際Ubuntu GNOMEはIBusを使っています。よってFcitxが使えるようにGNOMEを修正するか、IBusへ移行するかの判断が必要です。Fcitxサポート自体はそこまでたいへんではありません。Fcitxがいまだに Waylandに対応していないことを考えると、このままIBusへと戻る可能性があります。

またGNOMEは初回ログイン時の初期セットアップアプリを使って、日本語入力関連の設定を行います。現在のところこの初期セットアップアプリを標準でインストールするかどうかは未定です。よって17.10ではインストール後に何がしかの日本語環境のセットアップを手動で行う必要が出てくるかもしれません。

さよならUnity

2017年6月1日、ubuntu-metaパッケージのリストからunityが削除され、gnome-shellが追加されました[注3]。これはUbuntuの標準のデスクトップ環境がGNOME Shellに変わったことを意味します。最後にこの変更が行われたときの、コミットログを翻訳しておきましょう。SD

「さよならUnity、長くも楽しい旅路だったよ。最初はUbuntu Netbook Edition用のvala+mutterなUnity0だったね。Unity1からUnity7にかけては、Compiz/C++で再実装し、Nuxを追加したもんだ。その道のりは楽しく、喜びや悲しみに溢れ、狂気に満ちていて……あぁ、よくクラッシュしたことも忘れていないよ！　シェーダーやオーバーレイスクロールバーにウィンドウが動かなくなる問題も覚えている。そういえば朝5時にリリースしたこともあったね？　そうそう、Unity2Dチームも君と歩調を合わせようとしていたよね！　関わってくれたすべての人に感謝をしよう。まだ残っている、すでに去ってしまった、すべての人に。」

注3） http://bazaar.launchpad.net/~ubuntu-core-dev/ubuntu-seeds/ubuntu.artful/revision/2531

Unixコマンドライン探検隊
Shellを知ればOSを理解できる

Author 中島 雅弘(なかじま まさひろ) ㈱アーヴァイン・システムズ

先月に引き続きsshについて解説します。sshの応用と関連するコマンドを見ていきます。

第16回 ssh(その2)

sshを使えば、途中インターネットを経由しつつ複数のプライベートネットワークを安全に結ぶことができます。また、リモートコマンドの実行だけでなく、SSHプロトコルによって、ファイルやディレクトリの安全なコピー、GUIプログラムの安全なリモート転送など、VPN基盤として活用できます。

少し高度なsshの使い方

sshは、接続元とターゲットシステムを安全に接続するだけでなく、別のサービス、別のシステムへ安全に接続できる環境を作れます。ここでは、sshを使って、2種類のトンネルを作る方法と、Xプロトコルの転送を紹介します。

STEP UP! sshの凄技1 ポートフォワード(ローカル転送)

ローカル転送は、リモートホストの指定のポートに接続すると、ターゲットホストの指定したTCPサービスポートに転送します。トンネル内(ローカル－リモート間)の通信は、sshによって暗号化されています(図1)。

```
ssh リモートホスト -L 転送するポート:ターゲットホスト:転送先のTCPプロトコルポート [-g]
```

転送するポートは、接続可能な使われていないポート番号を指定します。sshを実行するローカルホスト以外からの接続を許可するには、-gオプションを指定します。

STEP UP! sshの凄技2 ポートフォワード(リモート転送)

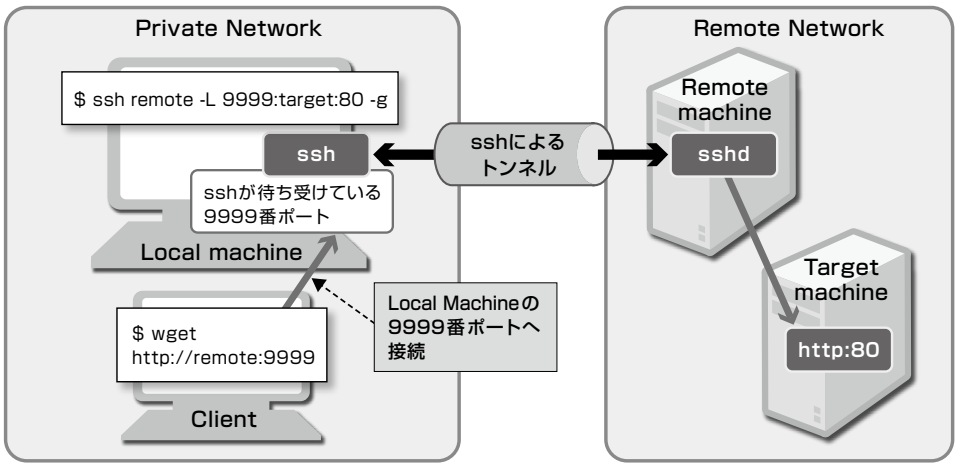
リモート転送は、プライベートネットワーク内のシステムへ、外部から接続できるようにトンネルを作る場合などに便利なしくみです(図2)。

```
ssh トンネルを作るホスト -R 転送するポート:転送先のホスト:転送先のTCPプロトコルポート
```

本来は、アクセスできないところへアクセスできるようにするしくみですから、使い方を間違うとセキュリティホールになります。-Rでのトンネリングは、リモートホスト外の未知のホストから、なんでも接続を受け入れていては危険です。sshdは、デフォルトの設定で、トンネルを作った状態では、転送するポートへアクセスできるのは、トンネルを作るホスト上からです。ですので、このトンネルを使うには、外部のホストから一度トンネルを作るホストにログインしてから、転送するポートに接続します。

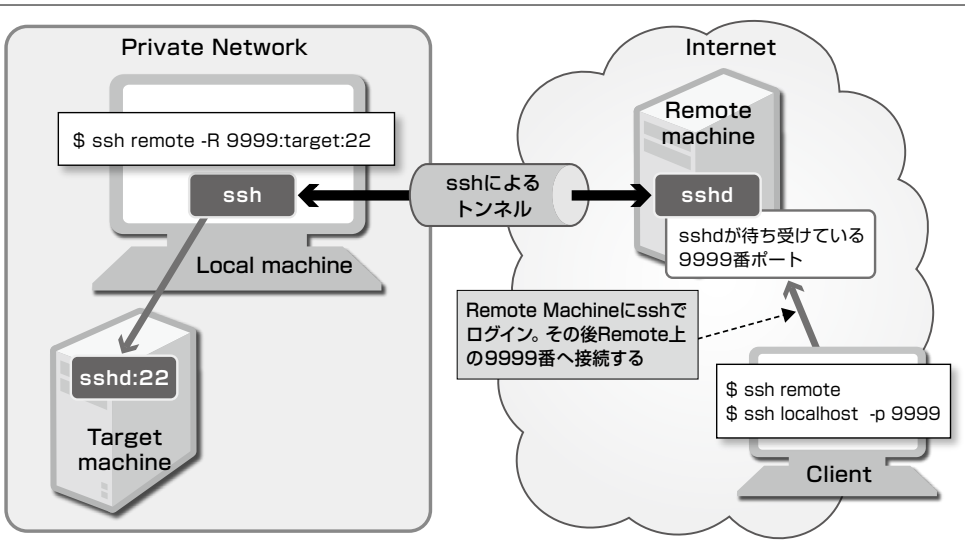
トンネルを確保しても、sshの接続がタイムアウトや何らかの理由で切断されてしまうと、

▼図1　ポートフォワード1　ローカル



ローカルマシンから、リモートマシンにssh接続してトンネルを作っておくことで、ターゲットマシン上の、別のTCPサービスへのプロキシとして動作する。-gオプションを付けることで、ローカルマシン外のクライアントからの接続を許している。

▼図2　ポートフォワード2　リモート



プライベートネットワーク内のマシンから、インターネット上のリモートマシンにssh接続してトンネルを作っておけば、プライベートネットワーク外のマシンから、プライベートネットワーク内のマシンにアクセスできるようになる。

接続しなおさなければなりません。このような環境を維持するためには、autosshコマンドを使うと、セッションが切れても自動で再接続してくれるので便利です。sshと基本的に同じ使い方ですが、モニターポートの指定などをするオプションが必要です。標準ではOSに入っていないので、brewやapt、yumで導入してください。

autossh起動の例
```
$ autossh -f -M 20000 -N -C -R 9999:localhost:22 her-ubuntu
```

-fオプションは、autosshをバックグラウンドで実行する。-Mオプションはモニタリングポートを指定する。-Nは、sshへのオプションでリモートコマンドを実行しない指定です。ここでは、ポートフォワードだけできれば良いので-Nが便利で

す。-Cは、通信を圧縮するオプションです。

STEP UP! sshの凄技3　X Serverとして使う

X Window Systemは、クライアント・サーバ型のウィンドウシステムです。アプリケーションを実行するホストと、表示するホストを分離できる、Unix環境の標準的なウィンドウシステムです。macOSは、独自のGUIで動いていますが、brewで導入できるXと共存が可能です。設定ファイルなどには、X11という名称が出てきますが、Xのプロトコルです。筆者が新人のころはまだ、X10を採用しているワークステーションもありましたが、X11はとても長い間安定的に使われています。

X Window Systemのサービスを、先出のポートフォワードのしくみに乗せて安全に転送することができます。便利なこのしくみは、-X(X11プロトコルのフォワード)、または-Y(信頼されたX11プロトコルのフォワード)を指定すると使えます。

X11に対応したアプリケーションには、xeyes、xcalc、xclockなどなど、たくさんのアプリケーションがあります。macOS brewの環境だと/opt/X11/binにある、xで始まる名前のコマンドで、Ubuntuなどの Linux環境だとだいたい/usr/bin/に入っています。GUIベースのファイルマネージャーなどもXで動作していれば、Xのフォワーディング機能が使えます。

リモートホストがサーバというイメージがありますが、X Window Systemは、アプリケーションを実行している側はクライアント、表示側がサーバです。

sshのクライアント側、sshdが動いているサーバ側の双方で、X11が使えるように設定しておく必要があります。クライアント側、~/.ssh/configの対象ホストに対して次を設定してください。

```
ForwardX11 yes
ForwardX11Trusted yes
```

次の例ではX11 forwardingに失敗しています。

```
失敗例
$ ssh my-ubuntu -Y
...略...
X11 forwarding request failed on channel 0
Last login: ...略...
```

この場合、リモートホスト側の(sshdの)設定が原因であると推察できます。/etc/ssh/sshd_configに、次の設定(が入っていなければ)を加えてみましょう。

```
X11Forwarding yes
X11UseLocalhost no
```

この設定を変更したら、sshdを再起動するか、親プロセスとなっているsshdにSIGHUPを送り[注1]ましょう。

```
$ ps ax | egrep sshd
 1011 ?      Ss     0:49 /usr/sbin/sshd -D
↑これが親プロセス
 5250 ?      Ss     0:00 sshd: masa [priv]
 5276 ?      S      0:00 sshd: masa@pts/24
...略...
$ sudo kill -HUP 1011
```

sshdのサービスを再立ち上げすることで、設定を新しく読み込ませる方法もありますが、サービスデーモンを完全に停止させてしまうよりも上記の方法が穏やかです。

macOSのsshd

macOSのsshdは、Linux系の環境と異なった形で起動されます。[システム環境設定]－[共有]で、「リモートログイン」をチェックすると、sshdが起動されるようになります。また、macOS独自のサービス管理のしくみであるlaunchctlから起動されるため、親になるsshdプロセスがありません。sshd_configを変更した場合でも、sshdを再立ち上げしたり、シグナルを送ったりしなくても、以降作られる新しいセッションには変更が反映されます。

うまくフォワードできたら、Xクライアントプログラムをローカルホスト上に表示してみましょう。Xを使うなど、ssh経由の転送量が多

注1) Linux系の環境でのsshdは、このシグナルを受けると、sshdは設定ファイルを読み直します。接続中のセッションは切断されません。

い場合、-Cオプションを指定すると通信データを圧縮します。サーバおよびクライアントも十分にマシンのパワーがあるのに、細い経路を経由する場合にも有効です。

her-ubuntu上のファイルマネージャーをローカルホストに表示する例
```
$ ssh -YC her-ubuntu nautilus --no-desktop &
```

Xを使わないなら、機密・性能を考慮して、-X、-Yオプションは指定しないでおきましょう。

macOS(XQuartz)とUbuntu 16.04でのX11 Forwardingトラブルシュート

デバッグオプション付きで、ssh -Y -v remotehostしたとき、次のエラーが出ることがあります。

```
debug1: No xauth program.
Warning: No xauth data; using fake authentication data for X11 forwarding.
```

xauthコマンドが実行できないために出ているメッセージですので、xauthが導入されていないなら、macOSならportやbrew、Ubuntuならaptなどで導入しておきます。導入されていても実行できない場合は、xauthの場所をsshが認識できていないことが原因かもしれません。

クライアント側の/etc/ssh/ssh_config、.ssh/configの"Host *"セクション(設定ファイルのフォーマットはmanを参照してください)で、たとえばmacOS Sierraでは、次のように適切なxauthの位置を指定する必要があります。

```
XAuthLocation "/opt/X11/bin/xauth"
```

macOSがサーバとなるなら、/etc/sshd_configにも同様に上記内容を記述します。

また、これとは別にリモートホストに/etc/ssh/sshrcもしくは.ssh/rcがあるとxauthを処理できないので、次のエラーが出てXが有効になりません(Ubuntuのsshd、macOSのsshd共通)。ですので、.ssh/rc、/etc/ssh/sshrcは作らないようにしておきます。

```
X11 connection rejected because of wrong authentication.
```

SSHプロトコルを使ったコマンド

scp——Secure CoPy

scpは、sshをデータ転送に使うことで、リモートシステムとのファイルコピーを安全に行えるコマンドです。scpを使えば、cpコマンドでファイルやディレクトリ(-rオプションを指定)をコピーするように、リモートシステムともやりとりができます。

```
$ scp ./himitsuno.txt my-ubuntu:~/secret/
```

SSHプロトコルによって経路の安全は確保できますが、ウィルスに感染しているなど、安全ではないファイルのコピーには注意しましょう。

rsync——a fast, versatile, remote(and local)file-copying tool——Remoteとファイルやディレクトリを SYNChronizeするツール

rsyncは、リモート、ローカルを問わず継ぎ目なくファイルやディレクトリを操作できる強力なプログラムです。ファイルシステムの階層構造をそのまま、あるいは一部を除外することもできますし、パーミッション、オーナー情報、タイムスタンプなども(望めば)そっくり複製できます。同期した環境では、以降は変更点のみをコピーすることもできますので、大量のファイルの同期が必要な環境でも活躍します。

sshが導入されている環境なら、SSHプロトコルでファイルのやりとりができるので安全です。rsyncdというサービスデーモンもありますが、最近のシステムならSSHを中心に運用するべきでしょう。rsyncは、挙動を制御するオプションがとてもたくさんあります。ここでは、よく使う単純な利用例を示しておきます。踏み込んだ使い方については、またあらためて紹介しようと思います。

リモートシステムからのバックアップで用いられる一般的な形式
```
例1 $ rsync -avz my-ubuntu:src/hana ~/backup/my-ubuntu
例2 $ rsync -auvz --rsh=/usr/bin/ssh --delete her-ubuntu.sonzaishinai.jp:/site /var/backup/Remote/her-ubuntu/http
例3 $ rsync -avz my-ubuntu:src/hana/ ~/backup/my-ubuntu
```

1つめの例は、ホストmy-ubuntuのホームディレクトリ以下の、src/hanaを階層構造をたどって、ローカルホストのホームディレクトリ以下

backup/my-ubuntuにアーカイブモード(シンボリックリンクや権限、所有者の情報などもそのまま)でコピーします。-zオプションは、転送時にデータを圧縮する指定です。2つめの例では、明示的にリモートシェルにsshを使うように指定しています。また、更新(-u：update)オプションを指定して、ローカルに新しいファイルがある場合にはスキップ、削除(--delete)オプションで、リモートからなくなってしまっているファイルをターゲットディレクトリ内からも削除しています。3つめの例は、1つめにそっくりですが、コピー元のディレクトリ名の後ろに"/"が付いています。こうするとコピー先には、hanaというディレクトリは作られずコピーされます(1つめの例では、コピー先のmy-ubuntuの中にhanaというディレクトリが作られます)。

そのほかSSHプロトコルを使えるコマンド

SSHプロトコルを使ったファイル転送プログラムには、先のscp以外にも、**sftp**コマンドがあります。また、リモートと情報のやりとりがあるコマンドは、経路を暗号化するためにSSHプロトコルを使って通信を行えるコマンドがあります。Git、Subversion(svn)などのリビジョンコントロールシステムは、SSHを始めさまざまなプロトコルが使えます。SSHの公開鍵認証を使えば、柔軟で安全なソースコード&ドキュメントの管理ができます。頻繁に新たなssh接続が必要な、こうしたリビジョンコントロールシステムを活用する場合には、パスフレーズの入力を省略する方法や、今回は説明しきれなかったssh-agentなどが役に立つことと思います。

sshのまとめとトラブルシュート

OS上でのさまざまな操作やプライベートネットワークにトンネルを作るなど、幅広くsshが活用できることを見てきました。裏返せば、悪意のある者に「sshをやぶられたら終わり」ということでもあります。ですので、IPで制限、堅牢なパスワードを用いる、堅牢な鍵を使うなどの徹底をしましょう。ssh_configやsshd_configでも、rootログインを許す、ポートフォワードを許すなどといったセキュリティを緩めるオプションの変更には細心の注意を払ってください。

OSやネットワークレベルでの接続を制限することも忘れずに。ファイアウォールや、sshがlibwrapをリンクしている[注2]、もしくは、TCP Wrapper経由で起動しているならhost.allow、hosts.denyの設定もきちんと確認しましょう。

一方接続ができないなどの問題解決作業では、設定の変更によって対象環境へのアクセスができなくなってしまうという事態を避けるため、できるだけ安全な実験環境を用意してください。そこで、ファイアウォールなどのアクセス制限を排除したりと、問題や課題を分離して原因を絞り込みます。本番環境で、設定を操作する場合には、稼働中のsshdとは別のポート(-p ポート番号)、別の設定ファイル(-f設定ファイル)で、新たなsshdを起動して動作を確認するのも良い方法です。

以下に、問題が生じた際に確認するポイントをいくつか挙げておきます。

接続できないとき

1. ネットワーク経路はつながっているか

ping、tracerouteなどを使って、ネットワーク経路が通じているか確認しましょう。

2. ファイアウォールなどでIP制限がないか

ファイアウォール、/etc/hosts.allow、/etc/hosts.deneyなどの設定を確認します。

sshで接続できない問題を解決するために、インターネットに露出しているサーバ環境でhosts.allowをsshd:ALLもしくは、ALL:ALLとする設定は危険です。せっかくのフィルタリング機能ですから、適切に接続元を絞り込み

注2) 最近の環境はほとんどlibwrapを使っています。

ましょう。

3. 接続元、接続先を変更してみる

どの接続先、接続元で生じる問題か、影響範囲を確認します。特定のホスト間でのみ生じている問題なら、前述の1番、2番などのネットワークの設定を見直しましょう。

4. ssh接続リクエストが適切に処理されているか

サーバ側のログを確認できれば、ssh接続リクエストが適切に処理されているか確認できます。sshdをデバッグオプション(-d)付きで起動できれば、より詳細なメッセージが得られます(**表1**)。

クライアント側では、sshをデバッグオプション(-v)付きで起動して、詳細な情報を出力させてみましょう。sshの接続中なら~vでデバッグレベルを上げてみましょう。何かしら有益な情報が得られるかもしれません。レベルを下げるには、~Vです。

sshdもsshもデバッグレベルは、オプション(-d/-v)を1～3つ指定でき、この個数でデバッグ出力のレベルが異なります(デバッグレベル3でsshを起動する例：ssh -vvv)。

問題の生じている環境のほかに、きちんと動作する環境があるなら、両者のログを比較してみるのも良いでしょう。

また、sshのバージョンが異なると、暗号方式の違いなどによりうまく通信ができないことがあります。

5. sshの設定が期待どおりか確認する

新しいOpenSSH(6.8以降)であれば、-Gオプションを使って、ssh実行時にどの設定を使って接続に行くか確認できます。~/.ssh/configや/etc/ssh_configでの問題が見つかるかもしれません。

同様にサーバ側では、sshdを-Tオプション(テストモード)を使って確認するのも有効です。

▼表1　sshがログを出力する先の例

環境	ログ出力先
Linux共通	/var/log/syslog
Ubuntuなど	/var/log/auth.log
CentOSなど	/var/log/secure /var/log/messages
macOS	/var/log/system.log

今回の技術が活躍するところ

今回は、sshの応用とSSHプロトコルを使うツールを、少々高度な使い方や設定方法まで踏み込んで解説しました。ファイアウォールの設定や、サーバ側、クライアント側、外部のコマンド、名前解決のしくみ、使っているソフトウェアやプロトコルのバージョンなど、sshの環境を整えることは、難しく感じられたかもしれません。しかし、sshを使いこなせば、作業の効率アップだけでなく、リモート環境でも安全な通信によって、システム管理者はもちろん、開発者にとっても強力なインフラ＆ツールとなるでしょう。

ホストベース認証の設定方法は紙幅の都合で今回は掲載できませんでした、ssh(その3)として解説する予定です。

次回について

次回は、テキスト処理(その3)として、テキスト処理の大関、横綱、sed＆awkを探検する予定です。SD

今回の確認コマンド

【manで調べるもの
(括弧内はセクション番号)】
ssh(1), ssh-keygen(1), autossh(1), ssh_config(5), sshd_config(5), X(7), xeyes(1), xcalc(1), xclock(1), xman(1), ping(1), traceroute(1), scp(1), sftp(1), tail(1), sshd(8), ssh-keysign(8), hosts.equiv(5), rsync(1), nautilus(1)(Linuxのみ), hosts_access(5)(Linuxのみ)

Linux カーネル観光ガイド

第64回 loopデバイスを非同期ダイレクトI/O対応にする機能

Text：青田 直大　AOTA Naohiro

loopデバイスの非同期ダイレクトI/O対応

6月19日にLinux 4.12-rc6が出ています。この次が最後のRCとなりそうですので、7月始めのころにはLinux 4.12がリリースされているのではないでしょうか。個人的には、Linux 4.12のあとの、過去のバージョンのstableリリースを心待ちにしています。Linux 4.9以降において、md arrayをreadonlyにするとそのarrayへの操作が再起動までいっさいできなくなるバグにあたってしまい、その修正が4.12後にstableに下りてくるはずですので……。

今月は仮想マシンイメージファイルをmountするときなどに使われるloopデバイスを、非同期ダイレクトI/Oできるようにした機能を紹介します。

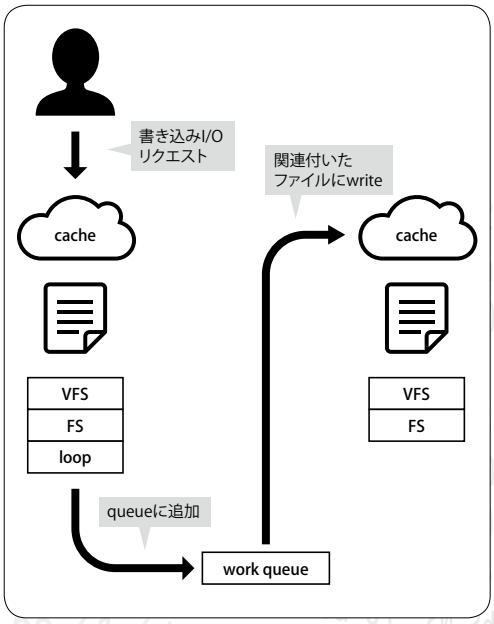
loopデバイスのしくみ

loopデバイスは、特殊なブロックデバイスの1つです。通常のブロックデバイスは、HDDなど実際のハードウェアに対応しています。一方でloopデバイスは、ほかのファイルやほかのブロックデバイスの上にマッピングを行うデバイスです。loopデバイスは、仮想マシンのディスクイメージや、ISOファイルをmountする際によく使われています。

ここで簡単にアプリケーションからの書き込みが、どのようにloopデバイスを通じて、マッピングされた下のファイルに書き込まれているのかを見ていきましょう(**図1**)。

loop上のファイルシステムに対するアプリケー

▼図1　loopデバイスのしくみ



ションの書き込みはVFS(Virtual File System)、ファイルシステムのコードを通して、loopデバイスへの書き込みリクエストになります。この書き込みリクエストは、loopドライバの**loop_queue_rq**という関数に渡され、ここからloopドライバによるI/O処理が始まります。

loop_queue_rqはシンプルな関数です。まず、渡されたI/Oリクエストに処理開始マークをつけ、loopデバイスがファイルに結びつけられているのを確認したうえで、I/Oリクエストを実際に処理するワークキューにリクエストを追加します。

このワークキューは、専用のカーネルスレッドで処理されます。各リクエストが読み込みなのか、書き込みなのか、あるいはフラッシュなのかなどによって呼び出される関数は違っていますが、書き込みの場合は**lo_write_simple**が呼び出されます。この関数は最終的に**vfs_iter_write**を呼び出します。これはカーネル内でVFSレイヤを使って書き込みを行う関数になります。あとは関連付けられているファイルシステムのコードにより、実際のデバイスへの書き込みが行われることになります。

二重のpage cache問題

loopデバイスが**vfs_iter_write**を使い、通常のバッファI/Oで書き込むと、同じデータに対してpage cacheを二重に確保してしまう問題が生じます。実際にその問題を、loop上に作ったファイルシステム内のファイルにデータを書くことで見てみましょう。"ext4.img"というイメージファイル上にext4のファイルシステムを作り、"/mnt/tmp"にmountします。**mount**コマンドにより、loopデバイスの"/dev/loop2"が"ext4.img"に対応付けられます。

図2のように**dd**コマンドで**foo**というファイ

▼図2　通常のI/Oでのpage cacheの挙動

```
$ sudo dd if=/dev/zero of=/mnt/tmp/foo bs=4k count=2
$ sync; sudo bash -c 'echo 1 > /proc/sys/vm/drop_caches'          # 既存のpage cacheを落とす
$ sudo dd if=/dev/zero of=/mnt/tmp/foo bs=4k count=2 conv=notrunc # 以下のコマンドを動かしながら実行
$ sudo ./perf-tools/bin/tpoint filemap:mm_filemap_add_to_page_cache
Tracing filemap:mm_filemap_add_to_page_cache. Ctrl-C to end.
              dd-23820 [004] .... 1203424.716139: mm_filemap_add_to_page_cache: dev 7:2
              ino d page=ffffea000711de80 pfn=1853306 ofs=0
              dd-23820 [004] .... 1203424.716162: mm_filemap_add_to_page_cache: dev 7:2
              ino d page=ffffea000711dec0 pfn=1853307 ofs=4096
           loop2-1724  [001] .... 1203424.716278: mm_filemap_add_to_page_cache: dev 8:3
           ino f8a867 page=ffffea00070c9540 pfn=1847893 ofs=140509184
           loop2-1724  [001] .... 1203424.716296: mm_filemap_add_to_page_cache: dev 8:3
           ino f8a867 page=ffffea00070c9580 pfn=1847894 ofs=140513280
(..中略..)

 # ddのcacheについて調べる
$ ls -l /dev/loop2
brw-rw---- 1 root disk 7, 2 Jun 12 11:34 /dev/loop2
$ ls -li /mnt/tmp/foo
13 -rw-r--r-- 1 root root 8192 Jun  8 15:23 /mnt/tmp/foo

 # loop2のcacheについて調べる
$ ls -l /dev/sda3
brw-rw---- 1 root disk 8, 3 Jun  8 18:34 /dev/sda3
$ ls -li ext4.img
16296039 -rw-r--r-- 1 root root 107374182400 Jun 12 11:34 ext4.img
$ sudo /usr/sbin/filefrag -v /mnt/tmp/foo
Filesystem type is: ef53
File size of foo is 8192 (2 blocks of 4096 bytes)
 ext:     logical_offset:        physical_offset: length:   expected: flags:
   0:        0..       1:      34304..     34305:      2:             last,eof
foo: 1 extent found
```

ルに4KBのブロックを2つ書き込み、syncします。このときに、page cacheがどのように確保されるかをトレースしてみましょう。トレースには、perf-tools[注1]のtpointコマンドを使います。filemap:mm_filemap_add_to_page_cacheというトレースポイントを対象にします。追加されたpage cacheが、どのデバイス上(dev)の、どのinode(ino)のファイルで、どのoffsetのデータに対応しているか(ofs)をプリントします。

ddを実行することで、図2のようなトレースが得られます。まず最初の2行を見ると、ddのプロセスが、デバイス"7:2"上のinode 0xd(=13)のファイル上のpage cacheを確保していることがわかります。デバイス"7:2"は、"/dev/loop2"のことです。すなわち、"/dev/loop2"上のfooの分のpagecacheがここで作られていることになります。そのあとの2行を見ると、今度は"loop2"というプロセスが、デバイス"8:3"上のinode 0xf8a867(=16,296,039)のファイル上のpage cacheを確保していることがわかります。"loop2"はカーネルスレッドで、先ほどのloopデバイス内のworkerにあたります。デバイス"8:3"は"/dev/sda3"のことで、"ext4.img"を置いているデバイスになります。inode 0xf8a867は"ext4.img"のことです。すなわち、"ext4.img"の140,509,184からと、140,513,280からのそれぞれ4KBをpage cacheに入れていることになります。

では、"ext4.img"のこのoffsetにはいったいなんのデータが入っているのでしょうか？filefragコマンドを使ってファイル"foo"のデータの物理アドレス、すなわちここでは"ext4.img"上の位置について見てみます。physical_offsetの部分を見ると、"foo"のデータは34,304 × 4KB(4,096) = 140,509,184から8KBの領域に書かれていることがわかります。この領域は、loop2がpage cacheに入れていた領域と一致します。すなわち、loop2は"foo"に書いたデータ用にpage cacheを作ったということになります。

このように、loopデバイスが通常のI/Oを行うと、「loop上のファイルシステム部分」で1つ、「loopの対象ファイルへの読み書き部分」で1つというように、同じデータに対して二重のpage cacheを持ってしまうことになります。

どちらのpage cacheを使うか

通常は、ファイルシステムが動作するブロックデバイスを直接アプリケーションから操作することはありません。したがって、同じファイルに対して2つのpage cacheを持たれるのは単に無駄にしかなりません。したがって、2つのcacheのどちらか1つだけを使うようにすれば、page cacheをよりよく活用できます。

通常のI/Oではなく、ダイレクトI/Oを使えばpage cacheの使用を回避できます。アプリケーション側のI/Oと、loopデバイス側のI/Oと、どちらをダイレクトI/Oにしても、page cacheを1つ削減できます。ここで、loopデバイス側をダイレクトにするほうが良い理由が3つあります。

1つ目の理由は、loopデバイス、すなわちカーネル側でやってしまえばアプリケーションを書き換える必要がないという点です。一般にアプリケーションはダイレクトI/Oをしていないので、プログラムの書き換えなどでダイレクトI/Oさせる必要があります。一方で、どこにでもダイレクトI/Oを使うと、loopデバイス以外へのI/Oではかえって遅くなる場合があります。したがって適切にI/Oを行うには、I/O先がloopデバイスかどうかを意識しながら、通常のI/Oとダイレクト I/Oを切り替えなければなりません。このようにアプリケーション側での実装には面倒が伴います。

2つ目の理由は、userlandとkernelでのpageの取扱いの違いにあります。ダイレクトI/Oは、page cacheを使わずにメモリ領域と直接デバイスに対してI/Oを行います。もしI/Oの途中で、

注1) https://github.com/brendangregg/perf-tools

対象のpageがスワップアウトされ、そこにほかの内容が読まれてしまうと、I/Oの結果が変わるというとんでもないことになります。したがって、対象のpageが動かないようにする必要があります。userlandのpageの場合、これにはpageの“pin”留めが必要です。この作業にはそれなりのコストがかかります。一方で、kernelのpageの場合には、pinの必要はありません。

最後に、よりアプリケーションに近い方が細かいcacheの制御ができるという点があります。例として、先ほどのファイル“foo”を削除しているときのpagecacheの削除の動きを見てみましょう(**図3**)。**tpoint**で**filemap:mm_filemap_delete_from_page_cache**をトレースし、どのpagecacheが削除されるかをプリントします。**rm**でファイルを削除し、**sync**するとinode 0xd(=ファイル“foo”)のoffset 0,4096からそれぞれ4KBのpagecacheが落ちているのがわかります。つまり、ファイルが削除されれば、そのファイルに関するpage cacheは削除できます。その一方で、loop先のファイルに関するpage cacheはまだ削除されていないのに注目してください。このようにアプリケーション側のpage cacheを残す方が、より細かいcacheの制御を活かせます。

非同期I/Oの導入

3つの理由から、page cacheの重複を解決するには、loopデバイスのほうをダイレクトI/Oに書き換える方がよいとわかりました。しかし、単純に通常のバッファI/Oをやめて、ダイレクトI/Oを使うとパフォーマンスが低下することがあります。

▼図3　ファイル削除とpage cacheの削除

```
$ sudo rm /mnt/tmp/foo; sync  # 以下のコマンドを動かしながら実行
 # ファイル削除にともなって、そのファイルについてのpage cacheが削除される
$ sudo ./perf-tools/bin/tpoint filemap:mm_filemap_delete_from_page_cache
              rm-4418  [007] d... 144577.796204: mm_filemap_delete_from_page_cache: dev 7:2
              ino d page=ffffea000ef3d140 pfn=3919685 ofs=0
              rm-4418  [007] d... 144577.796214: mm_filemap_delete_from_page_cache: dev 7:2
              ino d page=ffffea000ef6ba40 pfn=3922665 ofs=4096
```

fioコマンドを使い、SSDに対して通常のI/Oとダイレクト I/Oを行い、そのIOPS(I/O Per Second)を比較してみましょう。I/Oパターンは、シーケンシャルreadと、ランダムreadを使います。結果は**表1**のようになります。

結果を見ると、ランダムreadでは通常のI/Oもダイレクト I/Oもそんなに IOPSに遜色がない一方で、シーケンシャルreadではダイレクトI/Oは通常の1/3程度のIOPSとなっています。どうしてこのような差が出てくるのでしょうか。

IOPSの差を解明するために、より詳しくI/Oレイテンシを見てみましょう。シーケンシャルreadのバッファI/Oにおいては、レイテンシの中央値が2μsec(95パーセンタイル値でも2μsec)であるのに対して、ほかのケースでは最小レイテンシでも40μsecほどかかっています。このレイテンシの違いは、シーケンシャルreadのバッファI/Oにおいては、page cacheが有効に効いていることに起因します。バッファI/Oの場合、ディスクに対して先読みがかかり、アプリケーションから要求された範囲より後をあらかじめpage cacheに載せておきます。すると、多くのI/Oは先読みで、page cacheに載っているデータを読んでくればよく、レイテンシが小さくなります。一方で、ランダムreadではpage cacheにhitしないため、あるいはダイレクトI/Oではpage cacheを使わないために、ディスクへのアクセスがどうしても必要となり、レイテンシの下限が大きくなります。

page cacheを使わないダイレクトI/Oで、

▼表1　シーケンシャルreadとランダムreadのIOPS比較

	シーケンシャル	ランダム
バッファI/O	66,053	7,801
ダイレクトI/O	20,447	7,947

IOPSをより改善することはできないでしょうか。実は上記のI/Oパターンは、現代的なディスクにおいては、ディスクを完全に使用できていません。現代的なディスクは、ディスク内部にもI/Oキューを持ち、一度に複数のI/Oリクエストを処理できます。たとえば、"/sys/block/sda/device/queue_depth"を見ると"31"とあるように、31個までI/Oリクエストを送り込むことができます。ところが、上記のパターンでは一度に1つのI/Oリクエストしかディスクに送り込んでいません。

複数のI/Oを同時に送り込むにはどうしたらいいでしょうか。1つの方法は、今の同期ダイレクトI/Oのまま複数のスレッドを使うことです。しかし、複数のスレッドを使うとコンテキストスイッチの回数も増加し、パフォーマンスの悪化を招きます。そこで、非同期I/Oを用いて複数のI/Oリクエストをディスクに送るようにします。ふたたび**fio**で実験してみましょう。**--enginge=libaio**として非同期I/Oを使い、**--iodepth=31**と最大31個までのI/Oリクエストを送り込んでみます。

結果は**表2**のように同期ダイレクトI/Oに対して、シーケンシャルで20,447から48,755、ランダムで7,941から43,790と改善を見せています。とくにランダムにおいては、同期バッファI/Oよりも速くなっています。ランダムI/Oの場合、pagecacheがあまり効かないので、非同期I/Oでリクエストの数を増やす方が効果的ということですね。逆にpage cacheの効きやすいシーケンシャルでは、まだ同期バッファI/Oの方が勝っています。iodepthの数を512などと増やすことでその差は縮まりますが、それでも勝つことはありません。

▼表2　非同期I/Oを用いた場合のIOPS比較

	シーケンシャル	ランダム
iodepth = 31	48,755	43,790
iodepth = 512	57,303	43,834

このように、ダイレクトI/Oでは非同期I/Oにして、ディスクに複数のI/Oリクエストを送るのが望ましいことがわかりました。loopデバイスでダイレクトI/Oを使う場合でも、非同期I/Oが使われるようになっています。

ダイレクトI/Oを有効にする

loopデバイスでダイレクトI/Oを有効にする方法を見ていきましょう(**図4**)。まずファイルシステムのイメージファイル"fs.img"をmountします。すると、**mount**コマンドが自動的に空いているloopデバイスをセットアップしてmountします。**losetup**コマンドを使うと、現在のloopデバイスの設定状況を見ることができます。先ほどの"fs.img"が"/dev/loop0"にセットアップされていることがわかります。ここで"DIO"のフィールドが0になっていることに注目してください。これがloopデバイスがダイレクトI/Oを使うかどうかを示すフィールです。このようにデフォルトではダイレクトI/Oを使わない設定になっています。

ダイレクトI/Oを使うには**losetup --direct-io**を使います。このコマンドで、そのloopデバイスでダイレクトI/Oを有効にできるかどうかが検査され、ダイレクトI/Oが有効になります。実際**losetup**で見ても、"DIO"が1に変わっているのがわかります。逆に無効にしたい場合は、**--direct-io=off**を使います。

▼図4　ダイレクトI/Oを有効にする

```
$ sudo mount fs.img /mnt/tmp
$ losetup
NAME       SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE           DIO
/dev/loop0         0      0         1  0 /home/naota/fs.img   0
$ sudo losetup --direct-io /dev/loop0
$ losetup
NAME       SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE           DIO
/dev/loop0         0      0         1  0 /home/naota/fs.img   1
$ sudo losetup --direct-io=off /dev/loop0
```

ここでダイレクトI/Oが有効にできるかの検査があると言いました。ダイレクトI/Oを使うためには、いくつかの条件があります。1つはloopデバイスとファイルのあるデ

バイスのブロックサイズの関係による制限です。ブロックサイズは**/sys/block/sda/queue/logical_block_size**などで確認できます。ダイレクトI/Oを行うためには、loopのブロックサイズが、ファイルデバイスのブロックサイズ以上である必要があります。そうでない場合、すなわちloopのブロックサイズがファイルデバイスのブロックサイズよりも小さいとどうなるでしょうか。アプリケーションからloopへの書き込みのとき、loopのブロックサイズの方が小さいために、ファイルデバイスに書き出すにはデータが不足することになります。この不足した部分は一度ファイルデバイスから読み込む必要があるので、結局バッファI/Oと同じことをしてしまうことになります。したがって、**図5**のようなケースではダイレクトI/Oを使うことができないのです。

2つ目の条件として、loopデバイスのファイルへのマッピング位置は、ファイルデバイスのブロック単位にアライメントされている必要があります。loopデバイスはファイルの任意の位置を先頭として、マッピングできますが、この位置のアライメントがずれていると、サイズが不足する場合と同様にダイレクトI/Oができなくなります(**図6**)。

非同期ダイレクトI/Oの効果

loopデバイスを非同期ダイレクトI/Oにすることで、実際にどのような効果が出るのでしょうか。今回のpatchの作者が測定した結果があるので、それを見ていきます。

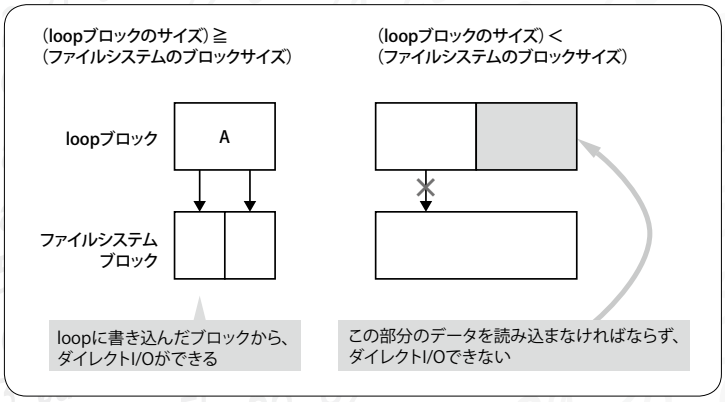
作者の実験では、4つの仮想CPU、2GBのRAMのKVM環境で、loopデバイスをSSD上に置いています。ここでSSDを使っているのは、その高いIOPS性能を活かして、IOPSで差が出るようにするためだと思われます。ワークロードとしては、fioを使い4つのプロセスでそれぞれ1.5GBのファイルに読み書きを行います。

まず、read/writeをそれぞれシーケンシャルとランダムとで行ったときのIOPSを見てみましょう。結果は**表3**のようになっています。このように、作者の実験によれば非同期ダイレクトI/Oにすることで、どのワークロードにおいても、既存の同期バッファI/OよりもIOPSが改善しています。一般には、readでは同期バッファI/Oの方が非同期ダイレクトI/Oよりもパフォーマンスがいいのに対して、この測定ではreadにおいても非同期ダイレクトI/Oの方が良いIOPSを出しています。これはダイレクトI/Oによって同一の内容に対するpage cacheが減ったことで、ほかの部分へpage cacheが割り当てられるようになったおかげだと考えられています。SD

▼図5　ブロックサイズの確認

```
$ grep . /sys/block/{sda,loop0}/queue/logical_block_size
/sys/block/sda/queue/logical_block_size:512
/sys/block/loop0/queue/logical_block_size:512
```

▼図6　ダイレクトI/Oができないケース



▼表3　loopデバイスを非同期ダイレクトI/Oにし、シーケンシャルとランダムでread/writeする

	read	ランダムread	write	ランダムwrite
同期バッファI/O	113,811	8,015	106,978	67,442
非同期ダイレクトI/O	125,040	8,136	111,376	67,811

August 2017

NO.70

Monthly News from jus

Japan UNIX Society

日本UNIXユーザ会　http://www.jus.or.jp/
りゅうちてつや　RYUCHI Tetsuya　ryuchi@ryuchi.org

シグナルもsedもこわくない？　シェル芸勉強会

今回は4月に実施したシェル勉強会について報告します。

シェル勉強会／シェルワンライナー勉強会

■jus・USP友の会共催 シェル勉強会／シェルワンライナー勉強会
【講師】今泉 光之（USP友の会）、上田 隆一（USP友の会）
【日時】2017年4月22日（土）10:00～18:00
【会場】さくらインターネット㈱ 西新宿セミナールーム

■はじめに

USP友の会と共催し、「第10回初心者向けなのかと百条委員会化する午前のシェル勉強会／第28回基準値を超えるシェル芸勉強会」を開催しました。

今回は28名の参加がありました。会場は西新宿にあるさくらインターネットのセミナールームでした。今回も有志によるサテライト会場が大阪と福岡に設けられ、ネットワークで接続されて開催されました。午前の部は今泉さんの講義と上田さんによる実演、午後の部は、上田さんが講師を担当されました。その後、懇親会とLT大会が同じ場所で開催されました。

■午前の部

午前の部はスクール形式にて、今泉さん（**写真1**）からUNIXで使われているシグナルをテーマにした勉強会がありました。自己紹介があったのち、シグナルの目的、種類や特性などの説明から始まりました。おもに使われるシグナルにはどのようなものがあり、それらがどう使われているのか、シグナルを発生する／させるためのメカニズムの説明がありました。また、それら以外にはどのようなシグナルがあるのかという点も紹介されました。

その後、シェルスクリプトからシグナルを使う手段として、trap(1)やkill(1)を使い、プログラムなどを作成していく方法が説明されました。シグナルの中には、シェルスクリプトで作成しているときにはほとんど発生しないが、C言語などでプログラムを作成しているとよく発生するシグナルも多いとのことでした。また、ユーザが自由に使えるシグナルもあり、その一例としてGNU dd(1)でどのように使われているかという点が紹介されました。続いて、シグナルの中でも特権的な動作をするものについて、必要な理由や、使われ方が説明されました。最後に、初期のころのシグナル、中期のBSDシグナルなど、シグナルの発展の歴史について説明されて、全体をまとめられました。

このあと、お昼休みまでしばらく時間がありましたので、上田さん（**写真2**）と交代されました。上田さんは端末を使い、シェルスクリプトを作成、実行しながら、シグナルの使い方を実演しました。シグ

写真1　今泉光之 氏

ナルの種類を確認し、どのような動作が期待されるか解説しながらシェルスクリプトを作成／実行し、実際にどのような挙動が端末で見られるかということを試されていました。

■午後の部

午後の部は室内のレイアウトを変更して、5～6人ずつの班を作り、参加者同士で問題に取り組む形式で開催されました（**写真3**）。この形式は、お互いに疑問やヒントを出し合いながら進めていくことで、難しい問題でも楽しく取り組めていけると感じています。

講師からは、近況やシェル芸についての説明に続き、今回のテーマについての説明がありました。今回のテーマは、用意されたLaTeXの原稿からいろいろなデータを取り出すことがメインでした。これは、講師が書籍の執筆において、LaTeXで作成した原稿から編集に必要なテキストを作成するときに行った作業をもとに問題を作成されたそうです。

LaTeXは文書作成ソフトウェアとして長く使われていますが、XMLのような構造を持たないため、シェル芸で扱うには少し複雑なようです。そこで今回は汎用的なものは目指さず、どんな原稿に対しても使えるものではなく、示された原稿に対して正しい解答を求められればOKとするものとなりました。たとえば脚注を探す問題では、脚注の終わりは必ず「。」（句点）で終わることを前提として解答が作成されていました。LaTeXの文法では、脚注の終わりは必ずしも「。」ではありませんが、今回の勉強会ではこれでOKでした。

写真2　上田隆一 氏

問題は全部で8問あり、sedの基本的な使い方の問題から始まり、文書の中からほしいセクションを取り出したり、ファイルに保存したりするような問題が次々と出されていました。これらを参加者が協力して解いていきました。しかし、講師は事前に解答例を作成して準備されてきているはずなのに、会場でうまく解けないというハプニングがありました。

■ネットワークを通じて遠隔での参加も可能

それから、今回もネットワークを使って勉強会の様子が配信されていたことから、解答はTwitterでも寄せられていました。ネットワークを通じて同じ時間を共有することも楽しいですし、実際に会場に参加して生で意見を交換しながら進めていくのも楽しいものだと、今回も感じることができました。

なお、Twitterでは「#シェル芸」のハッシュタグで意見交換を行っています。勉強会が開催される日に関係なく多くの人が投稿／閲覧していますので、質問があればつぶやいてみると、詳しい人が答えてくれるかもしれません。

■懇親会／LT大会

いつもは無事に8問の問題が終わるのですが、今回は講師の解答例のハプニングなどちょっと気がかりなことを残しながらの終了となりました。終了後はさくらインターネットのご好意で、同じ会場でビアバッシュ形式での懇親会を開催しました。今回も自己紹介とLT大会が開催されました。LT大会も、発表者が公開できないとされたもの以外は、ネットワークで中継されました。SD

写真3　午後の部の様子

あなたのスキルは社会に役立つ

2011年3月11日の東日本大震災発生の直後にHack For Japanは発足しました。今後発生しうる災害に対して過去の経験を活かすためにも、エンジニアがつながり続けるためのコミュニティとして継続しています。防災や減災、被災地の活性化や人材育成など、「エンジニアができる社会貢献」をテーマにした記事をお届けします。

第68回

災害時の連携とオープンデータ

●Hack For Japanスタッフ　及川 卓也（おいかわ たくや）　Twitter @takoratta

5月26日と27日の2日間、両国にある国際ファッションセンターKFCホールにて、「災害時の連携を考える全国フォーラム」が開催されました。これは、災害時に支援を行う者同士が平時から連携を進めるための場として開催されているもので、今年が第2回目となります。

筆者は一般社団法人情報支援レスキュー隊（IT DART）の一員として、2日目に参加してきましたので、そのレポートを行います。

また、筆者は東京都が設置した「ICT先進都市・東京のあり方懇談会」の「公共データ活用分科会」にも技術者を代表する有識者として参加していますが、それらの活動の中から見えてきたオープンデータ活用についての課題と可能性も特別コラムとしてまとめました。

災害時の連携を考える全国フォーラム

災害時には官民問わず、多くの支援団体や支援者が活動を行います。被災地入りする者もあれば、後方支援を行う者もいます。それぞれに支援内容は異なり、得手不得手もありますので、支援の効率性を高め、被災した地域の実情にあった活動を行うためには、各団体間の連携は不可欠です。そのような連携を平時から進めるために開催されるのが、「災害時の連携を考える全国フォーラム」です。

主催は特定非営利活動法人全国災害ボランティア支援団体ネットワーク（JVOAD）であり、災害支援を行う団体から構成される団体[注1]です。

第2回目となる今回のフォーラムは2日間かけて行われ、1日目のテーマは「過去の災害の教訓と、今の連携を学ぶ」で、2日目のテーマは「連携について、今後の課題解決を考える。」となっていました。筆者が代表理事を務めるIT DARTはおもに2日目を中心に参加してきました。

熊本地震から考える、支援のコーディネーション

2日目の全体セッション1はパネルディスカッションとなっており、行政関係者、支援団体、メディアとともに、IT DART理事の村上明子氏が登壇しました。セッションタイトルは「熊本地震から考える、支援のコーディネーション」です。

1年前に起きた熊本地震では、被災した住民からはTwitterなどのソーシャルメディアを通じて、困っていることなどが発信されましたが、そのニーズをどのように支援側は活用できたのか、今後に向けてどのように取り組むべきかが議論されました。

村上氏からは、Yahoo! JAPANから提供された検索データとして、「熊本地震」というキーワードとともに検索されたキーワードの変遷（4月と5月）や同じく4月と5月にTwitterで「熊本地震」とともにつぶやかれたキーワードが示され、その相違点などがそれぞれのメディアを利用する世代の違いにもあると考察されていました。

また、IT DARTがIBMより貸与を受けているIBM Watson Explorerを用いた、熊本地震関連の

注1　現在、22団体から構成されています。
http://jvoad.jp/#join

ツイートの分析も発表されました。それによると、リツイートを除く各避難所でのつぶやきは1避難所あたり全期間でも1～2程度であったことや、今回の地震で特徴的だった「車中泊」については恐怖や不安という感情とともにツイートされていたことがわかります（**図1**）。

また、品不足については、東日本大震災と熊本地震とで「水」に関連したつぶやきを時系列で追うことで、それぞれのニーズの違いも浮き彫りになりました（**図2**）。

このような分析を通じて、村上氏は「ソーシャルでわかることは限定的であり、すべての状況を把握することは不可能。しかし、現地調査をする前の情報把握など、有効な使い道がある。遠隔リソース（後方支援）などの1つとして活用していくと良いのではないか」と結びました。

災害時における支援の必要な情報の集約

午後には8つの分科会が開かれました。そのうちの1つがIT DARTの代表理事宮川祥子氏がコーディネーターを務める「災害時における支援の必要な情報の集約」です。

この分科会ではフォーラム全体のテーマともなっている、支援団体間の支援状況の可視化と調整を議論しました。支援状況の可視化と調整とは簡単に言うと、誰（Who）がどこ（Where）で何（What）をしているか、いわゆる3Wを可視化し、必要に応じて調整を図ることです。

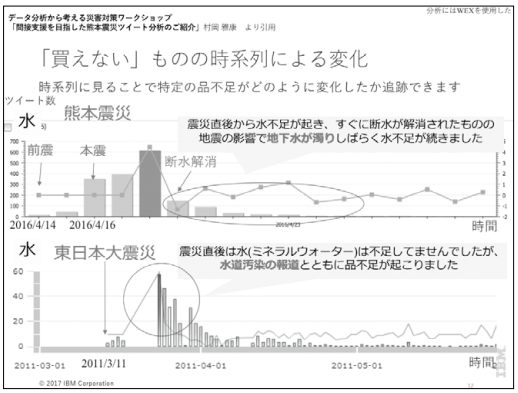
そんなことはすでにやれているのでは？と思われる読者の方もいらっしゃるかもしれませんが、災害発生時には情報が錯綜し、1団体内であっても正しい情報を共有することは簡単ではありません。とくに発災直後は状況が刻一刻と変化することもあり、情報をきれいにまとめ、他団体に共有する余裕などまったくないことも多くあります。

このような課題に対して、IT DARTは昨年のフォーラムで可視化のデモシステムを構築し、南海トラフ地震が起きたという想定でシミュレーションを行いました（**図3**）。

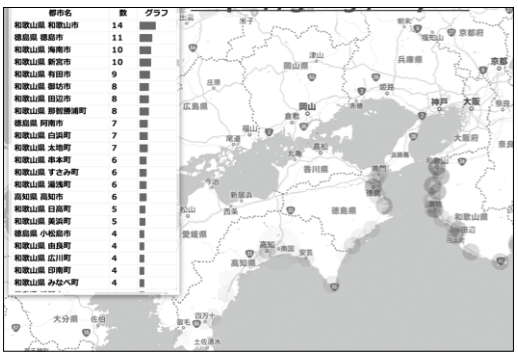
今回はさらに実用に近いものを構築し、分科会で

▼**図1　車中泊に関連したツイートの感情分析**

▼**図2　熊本地震と東日本大震災で「水」に関するツイートの傾向の違い**



▼**図3　昨年行った災害時の支援状況の可視化による連携・調整のシミュレーション。地図上に支援に入っている団体の量がヒートマップで示されており、枠内ではその実際の数値が見える**



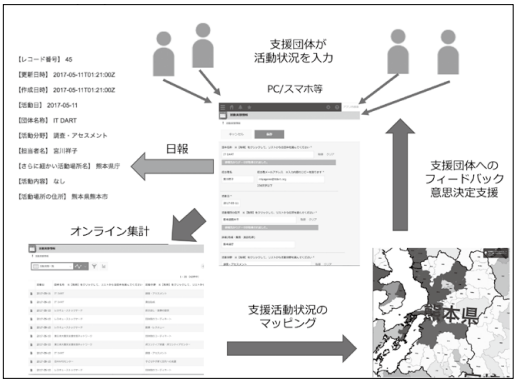
参加者に実際に試してもらいました。昨年はGoogleスプレッドシートのフォーム機能とGoogleマップを使ったものでしたが、今回はインターフェースにはサイボウズのkintoneを用い、地理情報空間システム（GIS）にはQGISを用いました。

参加者は実際に支援を行う団体の代表という立場として、スマートフォンやPC、またはタブレットから先ほどの3Wを入力してもらいました。結果はIT DARTスタッフがQGISで加工し、PDFで公開します(**図4**)。

kintoneを用いた理由はマルチプラットフォームに対応したユーザフレンドリーなシステムが構築できるためです。参加者からはスマートフォンアプリとしての提供も検討してはどうかという意見も出ましたが、現地の状況に合わせてシステムの機能や入力項目が変わっていく可能性もあるので、慎重に検討しています。

また、GISにQGISを用いたのは、オープンソースであり無料で使えることと、好きな出力フォーマットを選べ、情報もカスタマイズ可能なためです。Googleマップのように、入力した情報が即座にWebで見られることは魅力的ではありますが、実際の被災地ではネットが不安定なことも多く、紙で配布された情報のほうが重宝されることから、優先度が低いと判断しています。

▼**図4　分科会で用いた支援状況見える化システム。各団体が入力した支援状況(3W)を自動集計・地図情報として見える化した**



参加者からは、入力を各団体に任せるのは難しいのではないかという意見や、PDFのファイルサイズをもっと軽量にしてほしいなどの活発な意見が出されました。SD

Column　オープンデータ活用の課題と可能性

現在、日本は政府主導でオープンデータの活用を進めています。6月9日に閣議決定された「未来投資戦略2017[注A]」においても、建設分野の情報化としてオープンデータ化を進めることが明言されたのに加えて、分野横断施策として、公共データの利活用基盤や制度の構築、地方自治体職員を含む人材育成の強化と規制改革などが含まれています。

これを受けるような形で、東京都でも世界的なIT先進都市としてのデータの活用を進め、高度な都民サービスの提供を目指し、「ICT先進都市・東京のあり方懇談会[注B]」を設置しています。筆者はこの下の「公共データ活用分科会」に構成員として参加しています。ここでの議論内容は都市の情報セキュリティに関わる内容も含むため非公開となっていますが、筆者が都職員の方にお話する際に考えたことなどを説明することで、オープンデータ活用の課題と可能性が見えてくるのではないかと思います。

オープンデータはW3Cが定義した「5」を最高とするレーティングがあります[注C]。**図A**にあるように、PDFよりもExcel形式。Excel形式よりもCSV。CSVよりもRDF(Resource Description Framework)。そしてできるならば、LOD(Linked Open Data)が推奨されています。

国や自治体もこの推奨に則り、できるだけPDFやExcel形式でなく、CSVやXML(RDF)での公開を心がけるようになっています。しかし、筆者はあえて、「フォーマットにこだわらずに、まずはどんな形式でも良いので公開してください」とお願いするようにしています。

公開されさえすれば、フォーマットの変換はどうにかなります。Excelもプログラムから必要なデータを抽出することはできます。場合によっては、セマンティクス性が完全に失われてしまったCSVよりも扱いやすいこともあります。また、いざとなれば、どんなフォーマットであったとしても人力でデジタル化はできます。自動化や効率性を考えると人力で処理するの

注A　http://www.kantei.go.jp/jp/headline/pdf/seicho_senryaku/2017_all.pdf
注B　http://www.soumu.metro.tokyo.jp/13it/ictconf/index.html
注C　https://www.w3.org/DesignIssues/LinkedData.html

は避けるべきことですが、データが公開されずに利用できないことに比べれば、本当に価値のあるデータならば、そのフォーマットはなんであれ、まずは公開されることのほうが大事です。

データの公開に積極的になれない自治体などが多いようですが、その理由は何を公開して良いかわからないためであったり、データが悪用されることを心配しているためのようです。しかし、自治体が保持するデータはもともとは市民・住民のものです。情報公開法で請求すれば入手できるような情報や、紙媒体では公開しているような情報はオープンデータとしてすべて公開してしまって問題ありません。

オープンデータとして公開しても使われないのではないかという不安もあるようです。筆者も自治体の方に、どんなフォーマットでも良いから公開してしまいましょうと言ってはいるのですが、その結果、ちゃんと努力に見合うだけの利用があるかは保証できません。

たとえば、議会の議事録をテキストで公開してあるだけでも、テキストマイニングすればいろいろなことがわかります。議員の発言回数を時系列で見られたり、どのような内容の発言が多いのかも知ることができます。しかし、これにしても、国会ならば多くの方が興味を持つでしょうし、技術者もいろいろと解析したいと思うでしょうが、小さな自治体の場合、そもそも人口が国や都道府県に比べれば少ないので、どれほど興味を持たれるかわかりません。

つまり、オープンデータとして公開したデータが興味を持たれるかどうかは、そのデータのフォーマットなどにあるのではなく、そのデータの中身、もっと言うと地方自治体の場合ならば地方自治や行政にどれだけ興味が持たれているかしだいなのです。言い方を変えると、オープンデータへの興味は市民の政治参加意欲や行政と住民の距離のバロメーターということです。オープンデータを阻害するのは、行政側の非積極性だけでなく、市民の行政や政治への無関心にもあるのです。

また、オープンデータが活用されないのは、それがオープンデータだからという「オープン」の特性にあるのではなく、そもそものデータが活用されにくいものである可能性もあります。

理想としては、地方自治体の内部、すなわちクローズドな環境の中でかまわないので、データの利活用が進んでいることです。A部署のデータがB部署やC部署でも活用されているならば、このA部署のデータは確実に「使える」ものとなっているはずです。これを公開できるかどうか判断し、公開できるようならば、「使えた」実績のあるデータがオープンデータ化されることになるのです。

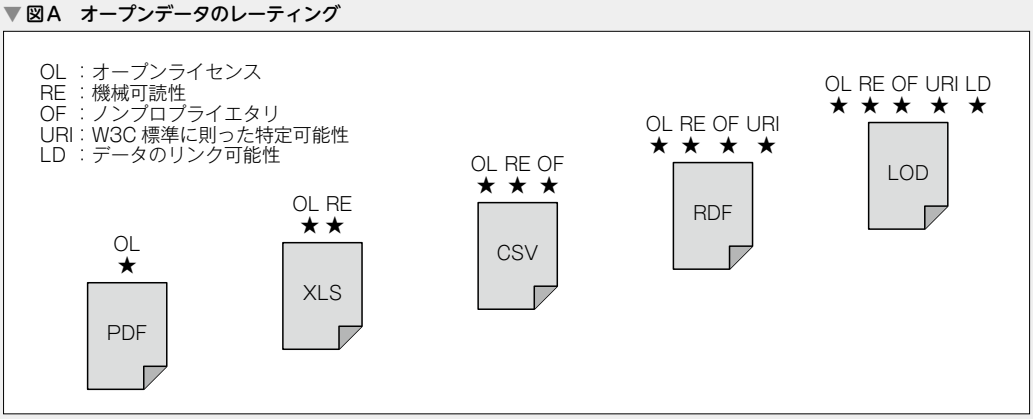
しかし、実態は“オープンデータとして使うために”データが生成されています。使われた実績のないデータが公開され、そしてやはりどこか使いにくいため使われない。使われないデータが公開され続け、市民の税金が無駄に使われ続ける。そのような事態を避けるためには、自治体内部のIT化を進め、自分たちが活用できているもののうち公開できるものを公開していくというアプローチに変えるべきです。

以上が筆者が東京都の活動やそのほかの自治体職員などとの意見交換などをするなかから見えてきた課題です。

これらの課題は、ただ、裏返せば可能性そのものです。市民の政治参加の話も、オープンデータを活用したワークショップなどを開くことで、オープンデータを契機に進むでしょう。自治体内のIT活用も、オープンデータとして用意するデータを、同時に自治体内で活用するように進めていけば良いでしょう。

オープンデータの公開は利用者視点で行うべきですが、その利用者としての市民の参加を促すとともに、自分たちも利用者となっていけば良いのです。

▼図A　オープンデータのレーティング



温故知新
IT むかしばなし

第68回

表計算ソフトウェア ～簡易言語からMultiplan、Lotus 1-2-3へ

速水 祐（はやみ ゆう） http://zob.club/ twitter @yyhayami

はじめに

Googleスプレッドシートなどのクラウド上のデータ管理が行えるシステムへと発展してきた表計算ソフトウェアですが、1980年代のパソコン黎明期には、まだ簡易言語とも呼ばれていました。日本においては、IBM-PC版に遅れること3年、日本語版Lotus 1-2-3（ロータス ワンツースリー）が発売されるまでは、一般ユーザにはあまり馴染みがないアプリケーションだったのです。今回は、この表計算アプリケーションの歩んで来た道のお話をしましょう。

簡易言語の登場

1980年、ソード社では独自のパソコンを発売しており、その上で動作する簡易言語PIPS[注1]を発表します。当時のパソコンでは業務用のソフトウェアは、BASIC言語を駆使して、担当者が大きな労力をかけて自分で作成することが多かったのです。

そこで登場するのが簡易言語です。画面上で表データを作成し、そのデータをコマンドや簡易な計算式で計算させ、その操作手順を記録すればプログラム作成となります。PIPSは、その発想と機能から注目を集め、ソード社製のパソコンと共に売り上げを伸ばしました。しかし、ソード以外のパソコンでは使えなかったため、大きく普及することはありませんでした。

表計算ソフトウェアの進化

海外では、1979年に登場したPersonal Software社（後のVisi Corp社）のVisiCalc（ビジカルク）が、大きく普及していました。VisiCalcは、表計算ソフトウェアの原型となる非常に革新的なソフトウェアで、パソコンが実用的に利用できることを証明して広く使われるようになりました。VisiCalcは、MOS Technology社のMOS 6502（Apple IIのCPU）用のアセンブリ言語で作成されており、その高速性と高機能性で、Apple IIのキラーアプリとなり、売り上げにも大きく貢献しました。

BASIC言語の販売では圧倒的な地位を築いていたMicrosoft社もVisiCalcの成功をみて、表計算ソフトの製作に着手することになります。

そこに登場したのが三菱電機の16bitパソコン、MULTI 16です。1981年12月に発表されたMULTI 16は、1982年に発売されたPC-9801より1年近く先行しており、16bitパソコンの原型[注2]となる優れたハードウェア設計でした。さらにDOS（Disk Operating System）としてCP/M-86を採用し、ROM BASICや漢字ROMを使わずにDOS上でフォントを読み込んでシフトJIS方式の漢字表示も実現していました。そしてこの高機能性を活かしたビジネスソフトウェアとして、Microsoft社が開発したばかりのCP/M-86上のMultiplan（マルチプラン）を採用するのです。それを追って、1982年10月にIBM-PC用のMS-DOS版Multiplanが発売されます。

Multiplanは、1983年にアスキー社からPC-9801版が発売され、日本でも表計算ソフトウェアが使われるようになります。しかし、Multiplanは汎用性が高

注1）現在Windows版The PIPSがフリーソフトウェアになっています。http://www.toshiba-tops.co.jp/archives/pips/download.html

注2）MP-1605。Intel 8088（4.44MHz）、メインメモリ384KB、画面解像度は640×400ドット（ドットごとに8色）、5.25インチFDDを2台搭載。123万円。

くなるようにDOS上で動作する設計で、C言語で作成されていたため[注3]、画面書き換えやスクロールなどの処理が遅くなっていました。

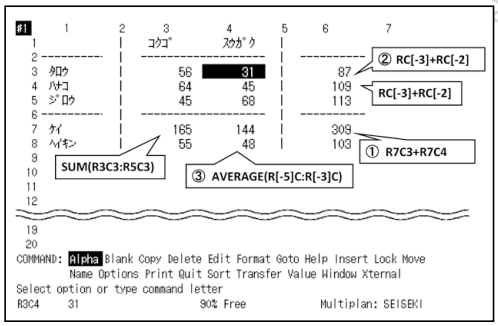
最大の表の大きさも、255行×63桁しか表現できなかったのですが、複数の表(シート)間でのデータリンクや計算ができました。

IBM-PCの世界では、Multiplanが登場した次の年の1983年にロータス社のLotus 1-2-3(以後1-2-3と略)が登場します。1-2-3は、IBM-PCのハードを活かして8086アセンブリ言語で作成されていたため、動作が高速で、その名前の由来となる、1(表計算)、2(グラフ作成)、3(データベース)の3つの機能に加え、マクロ機能も備える高機能な表計算ソフトウェアでした。そしてVisiCalcとApple IIの関係と同様に、IBM-PCのキラーアプリケーションとなり、さらに圧倒的に双方の売り上げを伸ばしていくのです。

日本においては、1986年にMultiplanがVer 2.0になり、4,095行×255桁と大きな表を扱うことができるようになり(メモリが許せば)、4倍以上の高速化とマクロ機能も追加となり、日本語版1-2-3の登場に備えることになります。

満を持して、1986年9月に日本語版1-2-3が登場します。日本語版はフロントエンドプロセッサ(漢字変換)に管理工学研究所の松茸を標準で付け、PC-9801についての高い技術力を持つ同社の力を得て移植開発が行われました。動作スピードも圧倒的であり、その高機能性から日本でも本格的に表計算ソフトウェアが普及していきます。

▼図1　Multiplanのセル表示



Multiplanの操作

Multiplanは、1-2-3やExcelと異なり、行番号も桁番号(Excelはアルファベット文字)も数値を用います(**図1**)。たとえば3行4桁のセルは「**R3C4**」と表します[注4]。7行3桁のセルと7行4桁のセルを加えた値は、「**R7C3+R7C4**」として「**R7C6**」に入力すると(①)、その結果が表示され、値が変わるとリアルタイムに計算結果も変化します。

今では当たり前のことですが、当時は画期的なことだったのです。セルの位置情報は相対的にも記述でき、自分の位置と同じ行で3桁前と2桁前のセルの合計を求めるためには**RC[-3]+RC[-2]**とできます(②)。

この記述方法を使うとコピーコマンドで下のセルに同じ計算式が設定されてもその行の値で正しく計算ができ、また並び替えを行っても式を修正せずに高速ソートができたのです。

その後の表計算ソフトウェア

1-2-3が普及すると、多くの業務をその上で実現することが試みられました。すると、表の大きさの限界が問題になってきました。画面の表示では8,192行×256桁まで可能なのですが、PC-9801の640KBのメモリの限界により、十分な表エリアが確保できなかったのです。そこで、メモリの拡張規格であるEMS(Expanded Memory Specification)メモリが使用されるようになっていきます。

パソコンはPC-9801、表計算は1-2-3、ワープロは一太郎が、標準になっていました。この状況は数年続きましたが、ここにWindowsの波が押し寄せます。Windowsアプリの開発にはMicrosoft社に一日の長があり、一太郎も1-2-3もWord + Excelの後塵を拝することになります。1985年Windows 95の発売と32bit化したMicrosoft Office 95(Word+Excel)によりOfficeの時代になりました。次はGoogle社などのクラウドを活用したシステムになっていくのでしょうか。SD

注3) 当時は十分なCPUパワーがなかったため、高速化を求める場合はC言語ではなくアセンブリ言語が使われていました。

注4) R1C1形式と呼ばれ、当時の1-2-3や現在のExcelでも設定すれば今でもこの形式のデータは扱えます。

NETGEAR ReadyNAS徹底運用

～大切なデータの保護に耐えうるか実力を試す！～

第1回 ReadyNASひとめぐり

データの保存先としてはもちろん、情報共有に使われる定番のアプライアンスといえばNAS（Network Attached Storage）。ひと口にNASといっても、現在ではさまざまな製品が市場に出回っていますが、中でも人気のブランドがNETGEARの「ReadyNAS」シリーズです（図1）。ここでは、その中から最新の機種を取り上げ、さまざまなシチュエーションでNASを運用し、その実力を試してみたいと思います。

Author 中山 一弘（なかやま かずひろ）

NASにはいろいろあれど

データの保存先としてNASが好まれる理由はさまざまですが、最も大きいのは、簡単に導入できる点と、要件に見合った製品が選びやすいことでしょう。たとえば、すでにファイルサーバを持っているような企業でも、プロジェクト単位やチーム単位で情報共有をしたい場合、共有フォルダが欲しくなります。しかし、そのつど情シスに依頼しなくてはならなかったり、要件や共有ユーザがコロコロ変わるようなケースでは、それすらめんどうです。

フットワークの軽いNASなら、要件に応じた容量の製品をネットワークに接続するだけで使えます。これは中小企業やSOHOにも当てはまるので、扱いやすく、製品も豊富なNASが選ばれるのだと思います。

筆者はライティングと企画／編集をメインにした編集プロダクションを経営しています。編集プロダクションとは、いわゆるコンテンツ制作業者で、あらゆる媒体へ向けて記事やWebページ、カタログやチラシといったコンテンツ作りのお手伝いをしている会社です。こういった仕事ですから、データは大小さまざまな形で毎日活用しています。クライアントや版元はもちろん、同じ制作業者ともデータのやりとりをするわけです。

かつてはCD-ROMにデータを焼いて郵送、なんていう時代もありましたが、最近はストレージサービスを使うことが多くなりました。現在ではクラウドサービスという名前に変わりましたが、要はネットワーク上のサービスを使ってデータのやりとりをするのが主流だったのです。

▲図1 人気のReadyNASシリーズ（画像は「ReadyNAS 628X」）。この製品を中心にNASの運用についていろいろと探ってみたい

それまでこれといって不便は感じなかったのですが、筆者の職業においても大きな変化がありました。それは動画コンテンツの台頭です。これまでは一番大きいデータでも写真程度だったので、画像満載の雑誌でも数百MBあれば十分でしたが、動画はそうはいきません。小さいものでも数GBと、これまでとは桁が違います（**図2**）。

こうした問題に対処するには、自社でファイルサーバを運用してしまうという方法があります。しかし、本格的なアプライアンスが買えるのはある程度の規模の企業でしょうから、多くの場合はNASをうまく運用していくほうがよいでしょう。実際に弊社でも、スタッフの間でNASがメインに運用され続けています。ただ、今動いているNASの容量は約2TB。現在のニーズを満たすには圧倒的に容量が不足していますので、そろそろ買い替えのタイミングでもあるわけです。

こうした状況にある中小企業は多いはずです。そんなみなさんに変わり、この誌面を借りてNASの効果的な運用を考えてみたいと思っています。

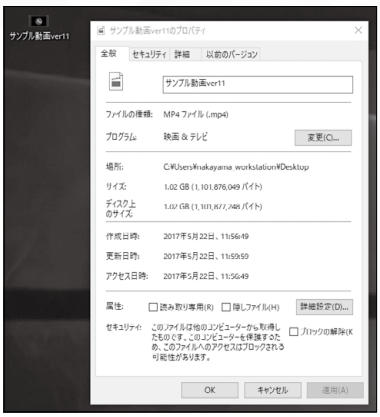
どんなNASを選ぶか、それが問題だ

NASといってもいろいろなタイプがあります。コンシューマ向けの製品で事足りるケースもあるでしょうが、将来的に見ると、便利なアプリケーションを使ったり、ディスクの追加や冗長性を高めたりと、管理機能が高いほどビジネスには有利です。

そのように考えたとき、現状で最も信頼できるのはNETGEARのReadyNASシリーズです。NASが今ほど浸透していないころから市場に製品を投入し続けていることもあって、信頼性は抜群です。ビジネスに応用できるデスクトップ型のNAS製品としても、2ベイから8ベイまでラインナップがそろっています(**表1**)。これだけあれば、ほとんどのケースで好みのタイプが選べるはずです。以下、各製品について簡単に紹介します。

ReadyNAS 620モデル

プロセッサにインテルXeonクアッドコアを採用し、メインメモリも8GBと大容量。120名のユーザがアクセスしてもストレスのないアクセス環境が提供できるモデルです。10GBASE-Tポートも2基搭載し、高速ネットワークとの親和性も抜群。本稿の評価機でもある8ベイの「ReadyNAS 628X」、6ベイの「ReadyNAS 626X」がラインナップしています。



▲図2 1GBの動画ファイル。5分程度のPVを完成させるのに、この手のファイルが山ほど生まれることになる。コストをかけずに安全に運用するにはNASが一番手軽な方法だ

ReadyNAS 520モデル

上位機種のReadyNAS 620の弟分ともいえるモデルで、プロセッサはインテルPentiumデュアルコア、メインメモリが4GBとスペックを絞り、コストパフォーマンスを高めた製品です。とはいえ10GBASE-Tポート2基を始め、筐体デザインも同じで、高い信頼性はそのまま80ユーザまで快適に使える仕様です。8ベイの「ReadyNAS 528X」、6ベイの「ReadyNAS 526X」、そして4ベイの「ReadyNAS 524X」が用意されています[注1]。

注1) ReadyNAS 524Xは、10GBASE-T×1となります。

▼表1 ReadyNASミニカタログ(デスクトップタイプ)

製品名	ReadyNAS 628X	ReadyNAS 626X	ReadyNAS 528X	ReadyNAS 526X	ReadyNAS 524X	ReadyNAS 428	ReadyNAS 426	ReadyNAS 424	ReadyNAS 422	ReadyNAS 214	ReadyNAS 212
ドライブベイ	8	6	8	6	4	8	6	4	2	4	2
対応トライブ	3.5インチまたは2.5インチSATA HDD/SDD(3.5インチドライブの搭載にはドライバ不要)										
最大容量	80TB (10TB×8)	60TB (10TB×6)	80TB (10TB×8)	60TB (10TB×6)	40TB (10TB×4)	80TB (10TB x 8)	60TB (10TB x 6)	40TB (10TB×4)	20TB (10TB×2)	24TB (6TB×4)	12TB (6TB×2)
プロセッサー	インテル Xeon D1521 (2.4GHz)		インテル Pentium D1508 (2.2GHz)			インテル Atom C3000 Quad Core Processor		インテル Atom C3338 Dual Core Processor		ARM Cortex A15 Dual Core Processor (1.4GHz)	
メモリ	8GB DDR4 ECC		4GB DDR4 ECC			4GB DDR4		2GB DDR4		2GB	
ネットワーク	10GBASE-T×2/ 1000BASE-T×2		10GBASE-T×2		10GBASE-T×1/ 1000BASE-T×1	1000BASE-T×4		1000BASE-T×2		1000BASE-T×2	
推奨同時接続数	120	120	80	80	80	40	40	40	40	一般家庭用	
市場参考価格(税込)	¥207,239	¥175,161	¥155,800	¥130,249	¥116,000	¥130,249	¥105,614	¥78,501	¥52,800	¥35,455	¥29,800

※上記価格は、HDDなしの「ディスクレスモデル」の価格です。2017年6月26日時点のアマゾンジャパンの販売価格を参考にしています。
※上記のほかにも、4ベイ、12ベイ、60ベイのラックマウント型ReadyNASをラインナップしています。

ReadyNAS 420モデル

こちらは40名程度のオフィスにぴったりなモデルで、コンパクトでスタイリッシュなボディが魅力の製品です。プロセッサはインテルAtom、メインメモリは2GBですが、使い勝手は上位製品と変わらず、扱いやすさと管理性の高さを兼ね備えているのが特長です。8ベイの「ReadyNAS 428」、6ベイの「ReadyNAS 426」(いずれもクアッドコア)、そして4ベイの「ReadyNAS 424」と、2ベイの「ReadyNAS 422」(いずれもデュアルコア)があります。

ReadyNAS 210モデル

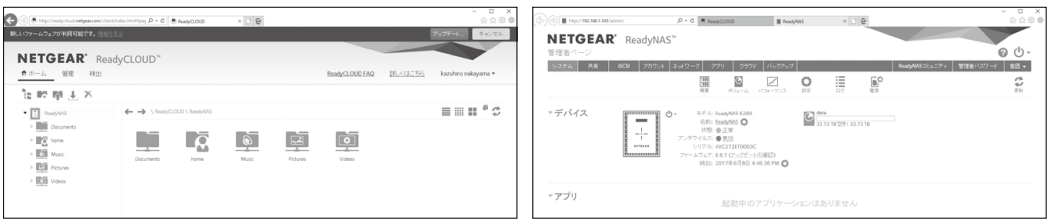
シリーズの中でも最もコストパフォーマンスに優れたモデルです。ARM Cortex A15デュアルコア、メインメモリ2GBと必要十分なスペックに加え、設置場所を選ばないデザインの高さが特長です。4ベイの「ReadyNAS 214」と、2ベイの「ReadyNAS 212」があります。

使いやすいコンソール周り

NASを選択する際に考慮したいのは管理ツールの扱いやすさです。これには各メーカーがあらゆる方法で特色を出していますが、決め手となるのはベースのOSと管理ツールのインターフェースでしょう。

ReadyNASへのアクセスや管理を実行するツールが、ブラウザ上で動作する「ReadyCLOUD」です。日常の利用状況の確認などはこの画面を通じて行うことになります。NASへのリモートアクセスに使用するほか、NASを直接管理する「ReadyNAS OS」にもここからアクセスしますから、何かと使う機会が多いツールでもあります。**図3**をご覧いただければわかるように、直感的にそれとわかるデザインのGUIで構成されているので、専門知識があまりない人でも安心です。

ファームウェアですが、本稿執筆時には「ReadyNAS 6.7.4」が最新バージョンになっています。もちろんファームウェアはつど更新されますから、導入された方は運用時にアップデートしてください。アプリケーションの導入方法やファームウェアのアップデートなどは次号以降で順次触れていきますので、ここでは割愛します。



▲**図3**　ReadyCLOUDの画面(左)とReadyNAS OSの画面(右)

COLUMN

なんと！ 購入前に管理ツールを操作できるぞ！

管理ツールは自分で触ってみないとなかなか感覚がわからないかと思います。そんな人向けに、NETGEARではReadyNAS OSの管理画面シミュレータ(**図A**)を用意しています。興味がある方はもちろん、操作に不安がある方もぜひ一度体験してみてください。購入前に管理画面を確認できるのでたいへん便利ですね。

▶**図A**　ReadyNAS OS6 - 管理画面シミュレータ
(https://www.netgear.jp/RNOS.html)

ReadyNASならではの強みも

アプライアンスとしての信頼性はもちろん、管理面でも評価が高いReadyNASシリーズですが、ファイルシステムや豊富なRAIDレベルといった部分でも強みを持っています。

ReadyNASが採用しているファイルシステムは「Btrfs(B-tree File system)」です。このファイルシステムの特長として、ファイル単位ではなく、ブロック単位でバックアップができる点や、高速アクセスが可能な点、さらには回数無制限のスナップショット機能や冗長性が高い「Copy-on-Write」機能が搭載できるといったメリットがあります。これによって、高速なバックアップやディスクスペースの効率化、自由自在なバックアップ運用やデータ損失のリスクを大幅に減らした運用などが期待できます。

また、NETGEAR独自の技術で「X-RAID」があります。これはディスクの本数に応じて最適なRAID構成を自動で行うもので、設定不要で冗長性や運用性が最も高い状態で管理できるのが特長です。ホットスワップへの対応はもちろん、段階的なディスクの追加などもできるので、ミニマムスタートで始めるNAS運用などにも向いています。

▲図4 「RAID Calculator」(http://rdconfigurator.netgear.com/raid/index.html)

X-RAIDは実際に運用するとその効果が実感できる、とても優れた機能だと思いますが、唯一の難点はRAIDを組んだあとのディスク容量が把握しづらいところでしょう。これに関してはNETGEARのサイトでシミュレータを公開しているので、事前にチェックしておくとよいと思います(**図4**)。

どんな状況にも対応できるNASの決定版

少し駆け足でしたが、ReadyNASシリーズの特長はご理解いただけたかと思います。このNASの良いところは、NASのOSに直接コマンドを打ち込んだり、アプリケーションを追加したりといった高度な使い方だけでなく、NASの知識がないような人でも簡単に扱える懐の深さにあるといえます。また、長期間の使用にも十分耐えられる信頼性の高さも特筆に値します。こうした特長は、ビジネスでの利用において、中小企業はもちろんSOHOレベルでの運用に最適であることを証明しているともいえるでしょう。

次回からはいよいよReadyNASシリーズの実運用をレポートしていきます。どんな内容になるか筆者も楽しみです。それではまた！SD

COLUMN

ベースのOSは「Debian」

OSに関してはほとんどのNAS製品がLinuxをベースとしています。ReadyNASシリーズも「Debian」を使っています(**図B**)。オープンソースベースですから、サードパーティが開発したアプリケーションを利用できるのはもちろん、ssh接続で実際にOSへコマンドを入力することもできます。

※SSH接続はメーカーサポート対象外

```
root@RN312:~# cat /etc/debian_version 8.7
root@RN312:~# uname -a
Linux RN312 4.1.30.x86_64.1 #1 SMP Wed
Nov 30 18:30:35 PST 2016 x86_64 GNU/Linux
root@RN312:~#
root@RN312:~# cat /proc/version
Linux version 4.1.30.x86_64.1 (jenkins@
blocks) (gcc version 4.7.2 (Debian
4.7.2-5) ) #1 SMP Wed Nov 30 18:30:35
PDT 2016
root@RN312:~#
```

▲図B OSバージョンを確認すると「Debian」であることが確認できる

うまくいく チーム開発のツール戦略

第9回 Subversionではだめなんですか!?

Author リックソフト㈱ 廣田 隆之

はじめに

この原稿を書いている時点で、衣替えの時期になりました。この春に入社された新入社員のみなさんの中には、そろそろ研修期間も終わり、実際のプロジェクトに配属されてコードを書き始めている方々もいらっしゃると思います。業務の傍ら新人教育の時間を捻出された先輩社員の方々もお疲れ様でした。

突然ですが、そんなみなさんに質問です。ソフトウェア開発を支える「三種の神器」については上司や先輩から教えていただいたでしょうか？ 上司や先輩のみなさんはこれらの開発ツールの重要性について説明されたでしょうか？ 私たちリックソフトの考える「三種の神器」とは、第一にソースコードのバージョン管理システム、第二に課題管理システム、第三に継続的インテグレーション(CI)システムです。

本記事では、バージョン管理システムに着目し、課題管理システムと連携しながらソフトウェア開発を進めていくプロセスを俯瞰(ふかん)します。次回以降では、CIシステムを活用してビルドやテストの自動化、デプロイの省力化を行う方法を紹介したいと考えています。

Bitbucket ServerとJIRA Software

バージョン管理システムや課題管理システムには多種多様な製品やサービスがあります。たとえば、バージョン管理システムとしてはSubversionやGitが広く普及しています。製品やサービスの例としては、Gitのクラウドサービスである GitHubやBitbucket、課題管理システムのRedmine、Trac、JIRAが有名です。

本記事では、Atlassianの商用製品であるBitbucket Server(以降、Bitbucket)とJIRA Software(以降、JIRA)を取り上げます。なお本記事で文中に注意書きがない場合はどちらの製品についても、みなさんがお持ちのサーバにインストールする形式のオンプレミス版を指します。

BitbucketとJIRAは、アプリケーションサーバとデータベースで構成されるWebアプリケーションなので、特別な専用のソフトウェアを必要とせず、Webブラウザがあれば利用できます。

追跡する

ソースコードのバージョン管理に専用のシステムを利用したほうがよい最大の理由は、いつ(When)、誰が(Who)、何を(What)、なぜ(Why)、ソースコードを変更したのかを記録するためです。ソースコードの変更点をこまめに記録しておくことで、チームのメンバーが変更点を確認できますし、数日後、場合によっては数年後の自分自身のためにもなります。たいていのプログラマは、ソースコードの変更点をすべて覚えておくことはできません。細かいデータの記録はコンピュータに任せて、私たちの頭の中は常に整理整とんしておきたいものです。

ソースコードの変更点を保存する際にJIRAの課題キー(バグやストーリーのひとつひとつ

を識別するための番号をJIRAでは課題キーと呼びます)をコミットログ(**図1**)に含めておけば、自動的にリンクが設定され、JIRAとBitbucketのどちらからも相互に参照できるようになります(**図2**)。

気軽に見る

Bitbucketは Webアプリケーションなので、ソースコードのコミット履歴やブランチ間の差分をWebブラウザで確認できます。ソースコードを手元のパソコンにダウンロードしてIDEやテキストエディタで開くといったためんどうな操作は必要ありません。

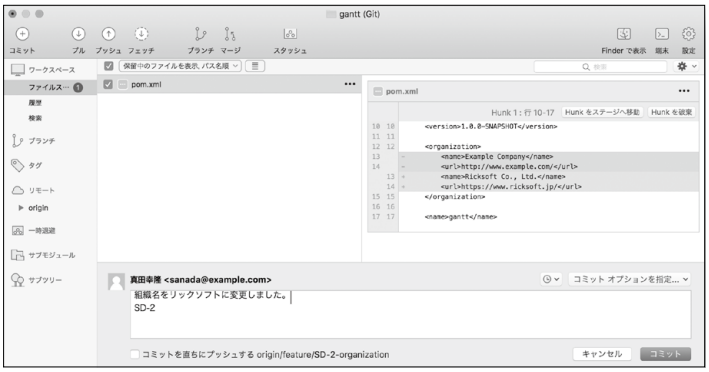
ファイルの内容が表示される画面では、特定の行を指定して表示させることもできるので、チャットなどでチームメンバーとURLを共有するといった使い方もできます(**図3**)。

また、タイプミスなどのちょっとした修正であれば、Webブラウザだけでも済ませられるので、たいへん便利です(**図4**)。

レビューして記録する

私たちがBitbucketをお勧めする理由の1つがプルリクエスト機能です。ソフトウェアの品質を高める活動の1つにコードレビューがあります。しかし、その効果やメリットをわかっていたとしても、レビューのためのしくみが整備されていないと実行は容易ではありません。レビューのたびにソースコードを紙に印刷したり、レビューのための時間と会議室を確保したり、レビューの記録を帳票として残したりと、組織

▼図1 SourceTreeでコミット



▼図2 JIRAの開発パネル

▼図3 ファイルの行を指定

によってさまざまな工夫をしていることと思います。しかし、往々にしてプログラマはこの手の作業が苦手なものです。

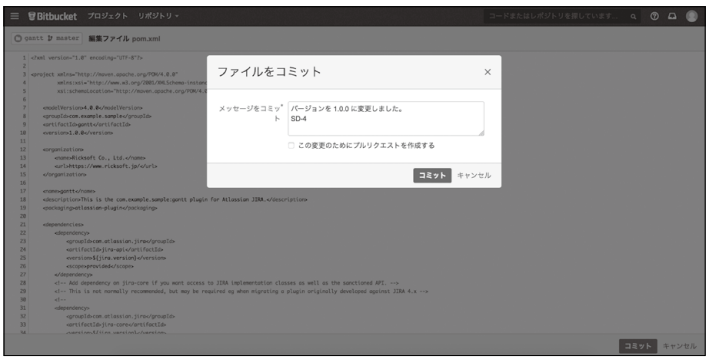
そこでBitbucketの出番です。Bitbucketのコードレビューは画面上で行います。Gitを使った開発では、機能の追加や不具合の修正のたびにブランチを作り(**図5**)、コードを修正したあとで分岐元にマージを行って開発を進めていきます。Bitbucketでは、マージするための承認プロセスをプルリクエストという機能で提供しています。

プルリクエストの画面では、ソースコードの差分やコミットの履歴を確認できます。レビュアは修正内容を確認し、問題がなければ承認を行います(**図6**)。コードには1行ごとにコメントを付けることができるので、画面上でコメントを追加してエンジニアどうしで議論を重ね、コードの品質を高めていくことができます。レビューの記録はBitbucketに残るため、レビューの履歴として後日活用することも考えられるでしょう。

これらの活動はオンラインでできますが、人間どうしで直接会話したほうがよいと感じた場合には、チャットやフェイストゥフェイスで会話し、その内容をJIRAやBitbucketに記録として残しておくことでさらに良いコードレビューになるでしょう。筆者も、コードを読んで理解できないときにはチームのメンバーに直接話を聞いて納得したうえで承認ボタンを押すということがよくあります。

マージには前提条件を設定でき、「2名以上のレビュアが承認していること」や「ビルドが成功していること」といった条件を指定できるの

▼図4　ブラウザ編集



▼図5　ブランチの作成

▼図6　プルリクエスト

で、プルリクエストのフローを厳格に規定することもできるようになっています(**図7**)。

その他の便利な使い方

プルリクエストは必ずしもマージを目的とする必要はありません。筆者たちがよく利用している「WIP(Work In Progress)プルリクエスト」と呼ばれるプラクティスを紹介します。

「実験的なコードをメンバーに共有したい」「現在自分が取り組んでいる仕事についてメンバーからアドバイスをもらいたい」といったとき、「WIP」という名前のプルリクエストを作成します(**図8**)。このプルリクエストを見たメンバーは、改善案をコメントとして付けたり、意見や感想をお互いに交わしたりします。このプラクティスは、チームの士気向上やソースコードのリファクタリングにもつながると考えています。

マージが目的ではないというよりも、むしろマージしてはいけないプルリクエストです。まれに「WIP」という名前を見落として、誤ってマージしそうになってしまうことがあるのが難点です。これを防ぐ改善点を目下検討中です。

おわりに

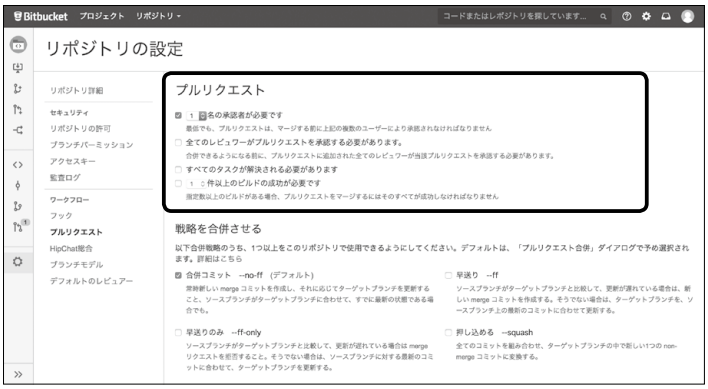
開発者にとって、日ごろ使い慣れた道具を変えることは大きなストレスであり、コストでもあります。しかし、GitやBitbucket、JIRAを活用することが、そのデメリットを補って余りあるパワーを与えてくれると筆者たちは考えています。

一気にすべてを変える必要はありません。新しく始まるプロジェクトのソースコードや、開発の傍らで作成するちょっとしたツール群など、気軽に始められるところから小さく始めてみてはいかがでしょうか。Bitbucketのクラウド版であれば5ユーザまではなんと0ドル、オンプレミス版であれば10ユーザ10ドルから始めることができます。

導入にあたってご不明な点や技術的サポートは、Atlassianプラチナソリューションパートナーであるリックソフトにいつでもお気軽にお問い合わせください。SD

※本連載の過去記事は技術評論社のWebサイト「gihyo.jp」でもご覧になれます。
http://gihyo.jp/ad/01/atlassian

▼図7　プルリクエストの条件設定



▼図8　WIPプルリクエスト

Special Interview

3年で300%急成長の

本誌で好評連載中の「うまくいくチーム開発のツール戦略」でお馴染みのリックソフト㈱が業務拡大につき各種エンジニアを募集中です。リックソフトは、Atlassian製品の販売でAPAC（アジア太平洋地域）ナンバー1の実績を誇ります。大手ポータルサイトを運営する職場からリックソフトに転職し、導入支援などで活躍している大崎健吾氏（**写真**）に転職の背景や現在の業務について詳しく聞いてみました。（編集部）

リックソフト㈱
大崎 健吾氏

エンジニアの仕事を支援する仕事に興味があった

――リックソフトに入社される前はどういったお仕事をされていたのですか。

大崎氏　もともと大手ポータルサイトを運営している会社でバックエンド側の開発を担当していました。利用していた言語は、Javaが5割、C／C＋＋が3割、残り2割がPHPといったところです。ただ、会社自体が大きく変わり、もっとも興味があった事業も外部のサービスを利用する形になったため、転職を考えるようになりました。もっとも興味があったのは検索エンジンです。学生時代に起業したときにも、ブログ検索のシステムを開発していました。これまでに10年ほど検索にかかわってきたのですが、転職活動を進めるときには、心機一転でまったく違う分野の会社に行ってみようと考えました。

――その中で最終的にリックソフトを選んだのはどうしてだったのでしょうか。

大崎氏　私の周囲にいたエンジニアはゲーム系の開発に転職するケースが多かったのですが、私自身は、ゲームを開発するよりもエンジニアを支援するプラットフォームに興味がありました。転職サイトなどを見て情報収集をしているときに、偶然リックソフトの存在を知りました。実は、以前にいた会社では10年ほど前から「Confluence」を使っていて、Atlassian製品については知っていました。リックソフトについて調べてみると、Atlassian製品を扱っていることがわかり、また家から近いこともあって（笑）、ちょっと行ってみようかと思って面接を受けました。すると、とんとん拍子で話がまとまったのです。

今の仕事にやりがいを感じ、多くの企業で導入を支援

――現在の業務内容を教えてください。

大崎氏　現在にメインとなっている業務は、Atlassian製品の導入支援です。効果的な使い方やカスタマイズまで含めてサポートしており、それにかかわるアドオンを自分自身で開発することもあります。

――業務の中でやりがいを感じるのはどういった場面でしょうか。

大崎氏　前職ではお客さまとじかにコミュニケーションをすることはなかったのですが、リックソフトでは導入を支援する中でお客さまから直接課題やお悩みをうかがいます。そうした課題を解決して、お客さまに喜んでいただいた瞬間はやりがいを感じます。たとえば、Confluenceや課題管理ソフトである「JIRA」（**図1**）を導入させていただいたお客さまが、1年後には積極的に活用されるようになっていて、それによって生産性が上がったなどポジティブな意見を伺えたときにはすごくうれしいですね。

ベンチャーがSE・PGを大募集!!

印象に残っているのは、ある大手製造業の企業でJIRAの導入を支援させていただいた案件です。その部署は新しいことに積極的に取り組んでいて、JIRAを導入する前はRedmineで課題管理を行っていました。しかし、ユーザ数が増えたことで限界が生じ、JIRAにスイッチすることになりました。そのプロジェクトをお手伝いしたのですが、その後その企業内でJIRAが評判になり、ほかの部署にも広がっていることに手応えを感じました。

▼図1　製造業などでも課題管理のためのツールとして導入が進むJIRA

――*実際にJIRAやConfluenceを導入する際、お客さまにはどのようなアドバイスをされるのでしょうか。*

大崎氏　よくお話しさせていただくのは、カスタマイズに凝り過ぎないことですね。標準のJIRAやConfluenceでは、業務にフィットしない部分をカスタマイズすることもありますが、やり過ぎるとコストやメンテナンスの手間がかさんでしまいます。基本的には、まず標準のシンプルな状態で使ってみて、業務にツールを合わせるのではなく、ツールに業務を合わせるくらいの意識があってもよいと思っています。せっかく業務を変える、効率化するという目的でツールを導入するのですから、あまり既存の業務にとらわれずに使いこなすことを考えるのも大切ではないでしょうか。

無理と言う前にまずやってみるエンジニアと働きたい

――*理想とするエンジニア像はありますか。*

大崎氏　変化を恐れず、興味を持ったことにはなんでもチャレンジしてみるエンジニアですね。口であああだこうだと言う前に、まずやってみる。それによって得られたものを製品にフィードバックし、お客さまや社会に対して還元できるエンジニアになっていきたいです。

――*リックソフトで一緒に働きたいエンジニアはどういった人でしょうか。*

大崎氏　リックソフト代表の大貫は発想が豊かで、アレをやりたい、コレをやりたいというアイデアが次々に浮かぶんです。要求に対して、すぐに無理だと考えてしまうのではなく、まずやってみたうえでどこが無理なのかを自分で検証する。そういった姿勢を持つ人が来てくれるとうれしいですよね。

――*導入支援という業務の範囲ではどういった人が求められていますか。*

大崎氏　大切だと考えているのは、お客さまのお話を柔軟に受け止められることです。特にリックソフトには、スピード感を重視するIT企業から、しっかり要件を固めたうえでステップを踏んで導入を進める金融業や製造業の企業など、さまざまなお客さまがいらっしゃいます。そうしたバックグラウンドを理解したうえでお客さまに対応していくことが重要です。その意味で、幅広い経験を持っている、あるいはリックソフトが提供しているサービスの世界を積極的に広げてくれるエンジニアと一緒に働きたいです。

――*本日はありがとうございました。*SD

Service

バッファロー、異常を検知したら即お知らせ、外付けHDDの故障予測サービス「みまもり合図」を開始

メルコホールディングスグループの㈱バッファローは7月5日、同社製外付けハードディスクおよびポータブルハードディスクの故障予測サービス「みまもり合図」を開始した。同サービス用のクライアントソフトウェア（Windows・Mac対応）は、無償で提供されている。

このサービスは、パソコンに接続されている外付けハードディスクのS.M.A.R.T情報をクラウドに蓄積し、状態を把握することでハードディスクの状態判定を通知するサービスだ。劣化進行時にはお客様番号がアラート通知され、その番号を専用サポート窓口へ連絡するだけで適切なサポートを受けられる。これにより、ハードディスク故障によるデータ消失を未然に回避することが可能になる。対象製品は、USB3.0/2.0接続の同社製外付けハードディスクおよびポータブルハードディスク（AV向け製品を除く）。

▲アラート通知例

CONTACT
㈱バッファロー URL http://buffalo.jp

Event

セキュリティコンテスト「SECCON 2017」、開催決定

6月29日、セキュリティコンテストイベント「SECCON 2017」のWebサイト（**https://2017.seccon.jp**）が公開された。

SECCONは、実践的情報セキュリティ人材の発掘・育成、技術の実践の場の提供を目的として、2012年から開催されている。パケットキャプチャやコード解析を駆使してクイズの答えを探したり、攻防戦を行ったりする「CTF（Capture The Flag）」のチーム戦がメインのプログラムとなる。2016年度に実施した「SECCON 2016」には99ヵ国から累計4,349人が参加、決勝大会には米国・韓国・台湾・ロシア・中国・ポーランド・コンゴ共和国・日本の14チームが参加し、ハッキング対決が繰り広げられた。

本大会と並行して、CTF初心者を対象とした「SECCON Beginners」や、情報セキュリティに興味のある女性向けの「CTF for GIRLS」などのプログラムも行われる。

●開催概要

主催	SECCON実行委員会
運営	㈱ナノオプト・メディア
日程	オンライン予選：2017年12月9、10日 ビジネスカンファレンス：2018年2月16日 決勝大会：2018年2月17～19日

CONTACT
SECCON 2017 URL https://2017.seccon.jp

Topic

ウェブルート、「ウェブルート脅威レポート 2017」を発表

ウェブルート㈱は6月15日、2016年を通じて収集したデータを分析した年次レポート「ウェブルート脅威レポート 2017」を発表した。ハイライトは次のとおり。

- IT企業を装うフィッシングサイトの数が、金融機関を装ったフィッシングサイトの7倍以上に。ターゲットとなった上位3社はGoogle、Yahoo!、Apple
- 感染のたびに自身を暗号化する「ポリモーフィック型」のマルウェアが猛威を振るう
- 2016年2月に初めて検出されたランサムウェアである「Locky」は、最初の1週間だけで40万件以上に感染
- 分析対象となった新規および更新モバイルアプリの50%近くを、悪意がある、またはその疑いがあるものとして分類、その総数は2015年の年間200万から2016年は1,000万近くに達した。最も増加したのは「アドウェア」
- 活発で悪意のあるIPアドレス数は3,300万件に上り、2015年からわずかに増加。検出を回避するために攻撃者がIPアドレスを変更するという前年からのトレンドが継続していることを示唆

CONTACT
ウェブルート㈱ URL https://www.webroot.com/jp/ja

Hardware

PFU、「Happy Hacking Keyboard Professional BT」に新色「白」を追加

㈱PFUは7月14日、Bluetooth接続のワイヤレスキーボード「Happy Hacking Keyboard（HHKB） Professional BT」に、新色「白」の3モデル（英語配列、無刻印（英語配列）、日本語配列）を追加した。

スペックそのほかは現行のカラー「墨」と変わらず、以下のようになっている。

・静電容量無接点方式のキー入力
・Bluetooth 3.0対応
・単三電池2本駆動（給電用USB micro-Bコネクタ付属）

PFUダイレクト（**http://www.pfu.fujitsu.com/direct/**）から購入でき、価格は27,500円（税別）となっている。

▲HHKB Professional BT 無刻印／白

HHKBシリーズは計算機科学者の和田英一氏（東京大学名誉教授）協力のもと、合理的なキー配列とコンパクトサイズを基本コンセプトとして1996年12月に発売され、2016年12月で20周年を迎えた。それを記念して特別サイト（**https://www.pfu.fujitsu.com/hhkeyboard/20th**）が開設されており、和田氏が厚紙で作成した「Alephキーボード」の模型や、アルミ削り出し・漆塗りの超高級モデルなど、HHKBシリーズの歴史を概観できる。

▲HHKB 20周年特別サイト

CONTACT
㈱PFU URL https://www.pfu.fujitsu.com

Service

TwoFive、なりすましメール対策のクラウドサービス「DMARC/25」を発表、なりすまし対策ポータル「ナリタイ」を開設

㈱TwoFiveは6月21日、なりすましメールへの対策を支援するサービス「DMARC/25」を発表し、その第1弾として、メール送信状況を解析するクラウドサービス「DMARC/25 Analyze」の提供を開始した。

DMARC/25 Analyzeは、DMARC[注1]で提供される膨大なXML形式の認証結果情報（DMARCレポート）を集計・可視化して解析し、Webベースのわかりやすいレポートとして表示する。なりすましの疑いのあるメール送信を検知した場合、レポート上に警告を表示し、管理者に通知する。このサービスを利用することにより、自社のドメインが不正利用されていないかを確認でき、なりすましの疑いがあるメール送信を迅速に検知できる。万一不正な送信に利用された場合でも、フィッシングメールの存在や内容を的確に把握できるので、自社のメールを受信する可能性のある顧客やビジネスパートナーに警告通知することにより、被害を抑止できる。

また同社は、なりすましメールによる被害撲滅のための活動の一環として、無料で利用可能なりすまし対策ポータルサイト「ナリタイ」（**https://www.naritai.jp**）を新設した。本サイトは、なりすましについての理解を深め、セキュリティ対策の企業導入促進やユーザの注意喚起を目的としている。なぜなりすましが起こるのかといった基礎的な情報をわかりやすく解説し、なりすましへの対抗策である送信ドメイン認証の設定方法などを、メールサービス別に説明している。また、自社メールシステムのドメイン名、IPアドレスを入力すれば、SPFの設定状況、RBL（DNSブラックリスト）の登録状況などを簡単に確認できる各種チェックのためのツールも提供している。

▲「ナリタイ」のコンテンツ例

CONTACT
㈱TwoFive URL http://www.twofive25.com

注1) 2つの送信ドメイン認証（IPアドレスに基づくSPFと電子署名に基づくDKIM）に失敗したメールを受信側がどう扱うべきかのポリシーを送信側で設定できるしくみ。

GitHub、カンファレンスイベント「Constellation Tokyo」を日本初開催

6月6日、日本で初となるGitHubのカンファレンス「GitHub Constellation Tokyo」が、TABLOID（東京都港区）で開催された。

カンファレンスは2つのトラック「GROW」と「BUILD」に分かれ、GROWではビジネスの成長や会社の枠を越えたエンジニアの活動を支えているGitHubの取り組みを、BUILDではGitHubのベストプラクティスといったテクニカルな話題を中心に講演が行われた。講演会場のほかには「Ask GitHub」というブースが常設されており、GitHubの技術メンバーに気軽に質問ができる。また、夜には懇親会の時間がとられ、GitHubを使うエンジニアたちが大いに情報交換できるイベントとなっていた。

基調講演は、ギットハブ・ジャパンのカントリーマネージャー藤田純氏によるあいさつに始まり、来日したGitHub Inc.のChief Business OfficerであるJulio Avalos氏がメインスピーチを行った。途中のゲストスピーカーとして、LINE㈱の長谷部良輔氏が登壇し、同社でのGitHubの活用事例を紹介した。

Avalos氏は冒頭、GitHubを使ってくれる人たちと、開発者としてだけでなく、グローバルシチズンとして、社会のさまざまな問題をテクノロジーを使って解決できるようになりたいという想いを述べ、より多くの人たちを巻き込んだ開発エコシステムとしての将来を語った。

また、「オープンかプロプライエタリか」という仕切りは、もはや消滅したのではないかと言及。その理由としてオープンソースへのコントリビュータ数を挙げ、上位にMicrosoft、Facebook、Googleといった大企業があるという現状を紹介。日本市場についても、Yahoo! JapanやKDDIの名を挙げ、そういう機運が高まっているのではないかとしたうえで、過去12ヵ月で売り上げが3倍に伸びている日本が同社にとって重要な市場であることをアピールした。

▲GitHub Inc. Chief Business Officer、Julio Avalos氏

CONTACT
ギットハブ・ジャパン合同会社 URL https://github.co.jp

「Google Cloud Next'17 in Tokyo」、開催

6月14、15日、ザ・プリンスパークタワー東京（東京都港区）においてGoogleのオフィシャルイベント「Google Cloud Next'17 in Tokyo」が開催された。日本のクラウド市場で猛進撃を続けるAmazonのAWS Summitがちょうど5月末に開催されたばかりで、今年はGoogleのクラウド「Google Cloud Platform（GCP）」に対しての強い意志が感じられた。

基調講演

シニアバイスプレジデントのダイアン・グリーン氏をはじめ、同社の幹部による基調講演では、GCPの最新情報に加え、「働き方改革」への取り組みが紹介された。もちろん人工知能・機械学習についての応用事例の発表もあり、非常にGoogleらしいGCPのアピールが行われた。

クラウドとAIの融合

個別のセッションではさまざまな企業のGCPの応用事例が紹介された。その中で特徴的だったのは、AIを使った自動応答システムの開発を進めているクラウドエース㈱の発表。同社は2008年からGCPに取り組み、日本で一番早くプレミアサービスパートナーを取得し、さまざまな企業でのクラウド移行・導入実績があるという。セッションでは、Googleが最近公開したapi.ai（人工知能API）を使用し、極めてシンプルに自動応答botを作るデモを行った。また、みずほヒューマンサービス㈱と開発を進めている事例として、GCPとAIを組み合わせた自動応答内線電話システムについても紹介を行った。このクラウドを利用したシステム導入で莫大なコスト削減が可能だという。

▲クラウドエース㈱　代表取締役社長　吉積 礼敏氏

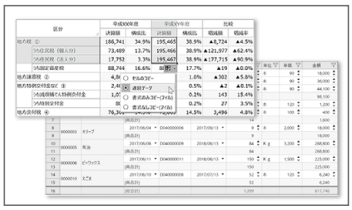
CONTACT
Google Cloud Next'17 in Tokyo
URL https://cloudnext.withgoogle.com/tokyo
クラウドエース㈱ URL https://www.cloud-ace.jp

Software

グレープシティ、
ExcelライクなUIを実現するWPF製品「SPREAD for WPF 2.0J」を発表

グレープシティ㈱は6月29日、.NET FrameworkアプリでExcelライクなUIを実現できる、Windows Presentation Foundation（WPF）向けコンポーネント「SPREAD for WPF」の最新バージョン「2.0J」を7月31日に発売する。1開発ライセンス価格は172,800円（税込）で、1ライセンスで3台のマシンにインストールできる。2.0Jのおもな新機能は次のとおり。

- フィルタリングドロップダウンにおいて、柔軟な絞り込み・検索が可能になった
- ドラッグフィルが強化され、テキストを含む数値の連続データ入力が可能になった
- アクティブセルの背景色や選択範囲の背景色の不透過度などを設定する際の複雑なXAMLの記述を簡素化可能になった
- 開発環境として、最新のVisual Studio 2017に対応した



▲SPREAD for WPF 2.0Jの画面例

CONTACT
グレープシティ㈱ URL http://www.grapecity.com

Service

A10ネットワークス、「Harmony Controller」を提供開始

A10ネットワークス㈱は5月29日、おもにアプライアンス製品、仮想アプライアンス製品として提供されていた同社のアプリケーション配信・セキュリティ・分析機能などを含むマルチサービスを、オンプレミスとマルチクラウド環境で一元管理できる「Harmony Controller」の提供を開始した。

Harmony Controllerは、イラスティックロードバランシング機能とWebアプリケーションファイアウォール機能をクラウドに特化させた「Lightning ADC」、ハードウェア・ソフトウェアで提供している従来の「Thunder ADCシリーズ」、オープンソースのロードバランサー「HAProxy」と連携し、オンプレミスからマルチクラウド環境全体で、アプリケーション配信とセキュリティの設定とポリシー、可視化を一元管理できる。

Harmony Controllerの提供形態としては、導入のハードルが低くコストパフォーマンスに優れた「SaaSモデル」と、データセンターやVMwareベースのプライベートクラウド、Amazon Web Services・Google Cloud Platform・Microsoft Azureといったクラウド環境内でスケーラブルなソフトウェアとして利用できる「セルフマネージドモデル」の2つがある。

CONTACT
A10ネットワークス㈱ URL http://www.a10networks.co.jp

Software

パラレルス、ツールボックス「Parallels Toolbox for Windows」を発売

パラレルス㈱は6月29日、アーカイブ機能や画面キャプチャ機能など、PCでの作業時間を短縮する多くのユーティリティツールが含まれたツールボックス「Parallels Toolbox for Windows」を発売した。

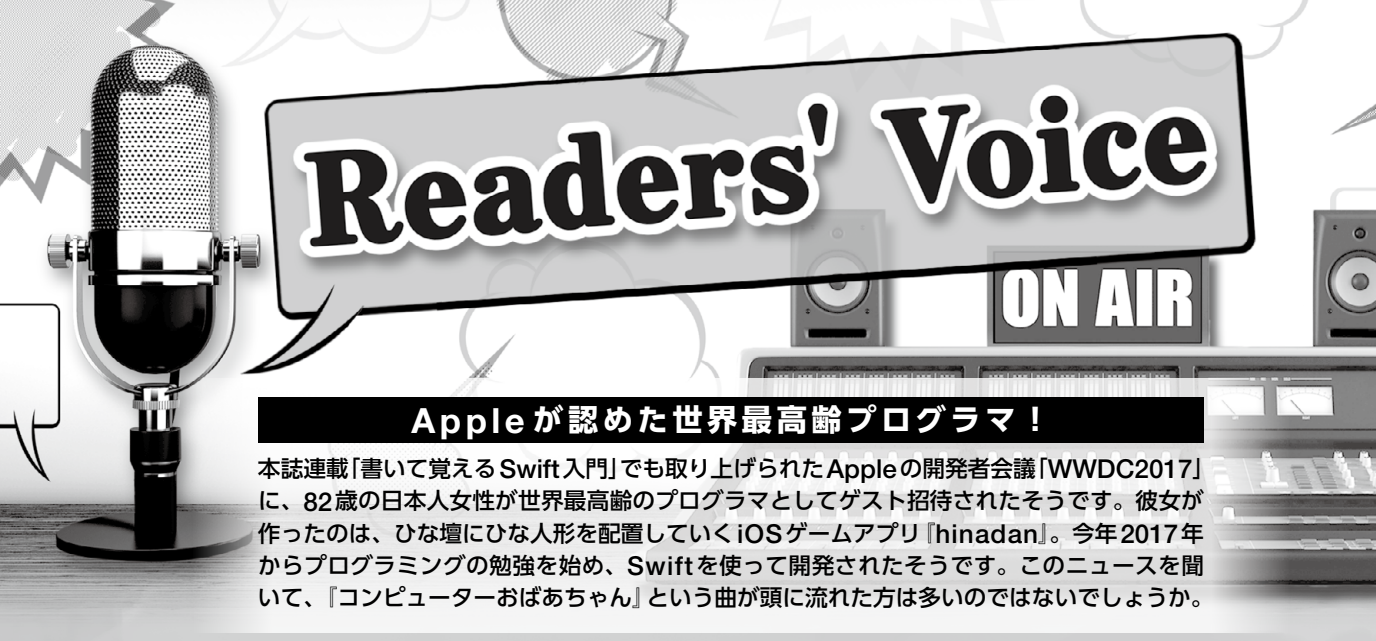
本製品はもともと「Parallels Toolbox for Mac」として販売していた製品のWindows版となる。おもな機能は表のとおり。

価格は年間1,000円／1ライセンスで、パラレルスのWebサイト（**http://www.parallels.com/jp/products/toolbox/**）からのみ購入できる。なお、ライセンスの有効期間内に追加の新機能やアップデーターが提供された場合は、無償で利用できる。

●Parallels Toolbox for Windowsのおもな機能

ジャンル	機能
ファイル	圧縮アーカイブの作成、ファイルの解凍／展開
動画	画面全体／指定範囲／ウインドウを録画。PCカメラによる動画の撮影・変換、YouTubeからの動画ダウンロード
画像	画面全体／指定範囲／ウインドウをキャプチャ、PCカメラによる写真撮影
音	PCマイクによる録音
制御	デスクトップの非表示、画面ロック、PCマイクのミュート設定、PCカメラのブロック、スリープモードOFF、通知・アニメーションの無効、複数のアイテムをワンクリックで同時に開く

CONTACT
パラレルス㈱ URL http://www.parallels.com/jp



Appleが認めた世界最高齢プログラマ！

本誌連載「書いて覚えるSwift入門」でも取り上げられたAppleの開発者会議「WWDC2017」に、82歳の日本人女性が世界最高齢のプログラマとしてゲスト招待されたそうです。彼女が作ったのは、ひな壇にひな人形を配置していくiOSゲームアプリ『hinadan』。今年2017年からプログラミングの勉強を始め、Swiftを使って開発されたそうです。このニュースを聞いて、『コンピューターおばあちゃん』という曲が頭に流れた方は多いのではないでしょうか。

2017年6月号について、たくさんの声が届きました。

第1特集　あなたのプログラミングを加速させるエディタ

プログラマに人気のテキストにエディタ、Vim、Emacs、Atom、Visual Studio Code（VSCode）を取り上げ、プログラミングに特化したカスタマイズ術を紹介しました。

基本のVimとEmacsに加えて、VS Codeを取り上げてくれていて良かった。
M.Nさん／奈良県

愛するIntelliJ IDEAも入れてほしい。
なおきさん／千葉県

普段VScodeを使っていますが、ほかのエディタの検討ができて良かった。
ぐれちゃんさん／東京都

普段はVimを使っているが、Ubuntu系のときはnanoを使っているので、nanoの特集も欲しかった。
ももんがさん／静岡県

Vim愛好家としては取り上げてくれるだけで吉。
hidewonさん／東京都

「WindowsでEmacsは茨の道」と、はっきり書いてあって良いです。
エポキシさん／神奈川県

紹介されていたVisual Studio Codeはこれまでまったく考慮したことがなかったのですが、初心者にもとっつきやすそうで良さそうでした。社員教育で導入してみようと思います。
犬棟梁さん／埼玉県

最近は宗教論争も多神教になったのですね。
山下さん／東京都

エンジニアさんの仕事道具として最初に名前が挙がるエディタということで、こだわりや愛着についての声が多く届きました。特集で取り上げたエディタのほかには「IntelliJ IDEA」「nano」「秀丸」の名前が挙がりました。

第2特集　今すぐはじめるPython

今最も注目されていると言っても過言ではないプログラミング言語Pyhtonについて、インストールや基本仕様といった入門から、「Jupyter Notebook」を使った機械学習まで、なぜPythonが人気なのかを実体験できる特集でした。

Pythonは初心者なので、おもしろく読ませていただきました。
小林さん／神奈川県

最近は何をやるにもPythonですね。今まで、コードを書きながらグラフ化したりということができることは知っていましたが、今回の特集で実際に試すことができました。これは便利ですね。Pythonにはまっていきそうです。
今井さん／千葉県

趣味プロでPythonを書いていると、職場でインデントにうるさくなる。
山崎亮さん／京都府

昔Pythonを使っていたのですが、最近機械学習関連で人気だということを知り、再び学ぼうかなと思っていたところだったので、今号の第2特集はタイムリーでした。2系と3系の違いが書かれていたのがとても良かったです。
地引さん／茨城県

いつの間にか、Pythonといえば機械学習、機械学習といえばPythonというほどのベストカップルです。そのほかの汎用のツール作りにも向いているようで、寄せられた声によるとほかの言語から移行されている方も多いようです。

6月号のプレゼント当選者は、次の皆さまです

①完全栄養食「COMP」
冨田大輔様（東京都）、とっぽ様（愛知県）、二関学様（東京都）、花子様（大阪府）

②コンパクトVRグラス
小堀大介様（埼玉県）、桑原直也様（神奈川県）、芳賀梢様（神奈川県）

③『PICと楽しむRaspberry Pi活用ガイドブック』＋ボード
渡邉真太郎様（東京都）

④『Infrastructure as Code』
石黒健太様（神奈川県）、jo7oem様（山形県）

⑤『The Art of Computer Programming Volume 4A』
松林諒様（京都府）、東雲様（栃木県）

⑥『Raspberry Piで学ぶROSロボット入門』
藤枝鋼様（東京都）、かず様（千葉県）

⑦『[改訂第3版]Apache Solr入門』
小澤亮太様（神奈川県）、藤原伸様（東京都）

※当選しているにもかかわらず、本誌発売日から1ヵ月経ってもプレゼントが届かない場合は、編集部（sd@gihyo.co.jp）までご連絡ください。アカウント登録の不備（本名が記入されていない場合、プレゼント応募後に住所が変更されている場合など）によって、お届けできないことがあります。2ヵ月以上、連絡がとれない場合は、再抽選させていただくことがあります。

一般記事　ハッシュ関数を使いこなしていますか？【前編】

ソフトウェア開発のさまざまな場面で利用されているハッシュ関数を深掘りする前後編です。前編では、ハッシュ関数の性質とおもな用途を紹介しました。

連想配列とハッシュの関係は知らなかったので、自分自身のコーディングに役立てようと思います。
ぴょうへいさん／大阪府

若い子だと言葉自体知らなかったりする。
うたさん／大阪府

ハッシュ関数の原理は知っていたが、使用することはなかったため勉強になりました。
いくさん／兵庫県

ハッシュのしくみ、重要性、用途がわかったところで、では実際に使ってみよう、という読者が増えた様子。「そんなところにまで使われていたとは」という声も多かったです。

一般記事　Windows Server 2016で構築する最新ファイルサーバ【前編】

Windows Serverの最新バージョン「2016」の新機能を紹介する前後編です。前編では、ディスククォータ、ファイルスクリーン、記憶域レポート、ファイルの自動分類、共有フォルダの一元管理などについて解説しました。

普段ほとんどWindowsに触らないため、新しい世界がわかりました。
n0tsさん／東京都

有償のソフトウェアで、業務でも使う機会はありませんが、むしろ好奇心を持って読めました。拡張子ベースのフィルタリングなど、おもしろい機能があると思いました。
斉藤さん／東京都

Windows Server 2012にようやく慣れてきたところなのですが、もう2016ですか。サイクルがずいぶん速く感じます。
永作さん／東京都

Linux＆OSS好きの開発者の方が多い本誌では珍しい、Windows Serverの記事です。大きな組織でのファイル管理・共有には、多機能でGUIでの操作に優れたWindows Serverがやはり便利ですね。

一般記事　mBaaSのしくみ紹介【3】

モバイルアプリのバックエンド開発を省力化できるmBaaS。その1つ「ニフティクラウドmobile backend（NCMB）」のしくみと使い方を解説する短期連載です。第3回では、Raspberry PiとNCMBでIoTドア監視アプリを作りました。

便利で、業務の効率化には最適なmBaaSのしくみがわかり、親しみが湧きました。
オミオさん／宮城県

CEDECでNCMBを見て興味を持っていたので、身近なテーマで良かった。
堕電さん／千葉県

as a Service系は各社いろいろ出していて、比較がたいへんです。
lipgtxさん／東京都

IoTの実践ということで、興味を持った読者が多いようです。デバイス、データ、通信と考えなければならない事柄が多いIoTでは、せめてバックエンドシステムはクラウドサービスに任せたいという方は多そうです。

コメントを掲載させていただいた読者の方には、1,000円分のQUOカードをお送りしております。コメントは、本誌サイト**http://sd.gihyo.jp/**の「読者アンケートと資料請求」からアクセスできるアンケートにてご投稿ください。みなさまのご意見・ご感想をお待ちしています。

次号予告

Software Design
September 2017

2017年9月号
定価（本体1,220円+税）
184ページ

8月18日発売

［第1特集］根底から理解していますか?

Web技術【超】入門

ブラウザもスマホアプリ開発も、案外知らないこと多いですよね!

■第1章　Webサーバと通信できますか?――ハンズオンで実際にやってみるhttpのしくみ
■第2章　ApacheとNginxの違いがわかりますか?――Webサーバ機能○×星取り!
■第3章　Web開発のOSS定番スタイルをふりかえる――Servlet、JSPからMVC、JavaEE、Springへ
■第4章　世界で高いシェア・IISでWeb開発――Visual Studioで一気通貫の高効率の実現とは
■第5章　スマホ開発の実際――ゲームからサーバ構築、クラウド利用の現場ノウハウ

［第2特集］黒い画面の使いこなし方

「tmux＆byobu」開発効率アップのターミナル改造術

※特集・記事内容は、予告なく変更される場合があります。あらかじめご容赦ください。

お詫びと訂正

以下の記事に誤りがございました。読者のみなさま、および関係者の方々にご迷惑をおかけしたことをお詫び申し上げます。

■2017年7月号
●P.16　アラカルト「読者プレゼントのお知らせ」プレゼント01の写真
「Aterm WG1200HP2」の製品写真が、「Aterm WG2600HP2」のものになっていました。

SD Staff Room

●このところ本を買っても積ん読が多くとてももったいない。スマホでFBなどチェックしているとすぐに読書の時間が奪われるからだ。それでも本を読むと他社さんの版面がとても新鮮に感じる。でも他の雑誌を読むと編集者目線になってしまう。企画の立案風景まで目に浮かぶ。これは職業病だと思う。（本）

●HuluがアップデートされてPS3で見られなくなった。TVが初期のプラズマハイビジョンなのでHDMI接続じゃなくてD端子接続なのが原因だが、NetflixもAmazon Premium Videoも問題なく見られている。ダウンロード再生もできないので見切りをつけた方が良いかも。それとも4KTVに買い換えるか。（幕）

●おなかの中身はすっからかんに出て行くのに、仕事はどんどん溜まっていくという3日間を過ごしました。インフルエンザなみの熱と離れられないトイレ。「おなかのカゼ」あんたそんな生やさしい感じじゃないよ。みなさまもウィルス性胃腸炎にはご注意を。そしてすべての関係者にお詫びと感謝を。（キ）

●私はFacebookもTwitterも使っていますが、人の投稿を見るだけで、自らはほとんど投稿しません。特定の人に伝えたいことはあっても、不特定多数に伝えたいことはあまりないものですね。私は結局、メールが一番使いやすいのですが、1対1で相手に合わせたやりとりができるからかもしれません。（よし）

●湯浅政明監督の大ファンで、4月公開の『夜は短し歩けよ乙女』、5月公開の『夜明け告げるルーのうた』は、どちらも素晴らしかった……。独特なデザインのキャラクターがカラフルな画面中を動き回って、一瞬も目が離せませんでした。同監督のシリーズ物だと、『カイバ』『ケモノヅメ』がお勧め!（な）

●最近の通勤時のお楽しみは、毎年同じ場所で巣作りする燕さん。今年もまた最寄駅の周りを燕の親鳥が飛び交い、毎日すくすくと元気に成長していく様子を見せてくれました。今では翼の羽もすっかり揃って飛ぶのも上手になり、もうすぐ今年の雛鳥たちの巣立ちの日が近づいています。無事に羽ばたいていってくれますように。（ま）

ご案内
編集部へのニュースリリース、各種案内などがございましたら、下記宛までお送りくださいますようお願いいたします。

Software Design 編集部
ニュース担当係

［E-mail］
sd@gihyo.co.jp

［FAX］
03-3513-6179

※FAX番号は変更される可能性もありますので、ご確認のうえご利用ください。

Software Design
2017年8月号

発行日
2017年7月18日

●発行人
片岡　巌

●編集人
池本公平

●編集
金田富士男
菊池　猛
吉岡高弘
中田瑛人

●編集アシスタント
根岸真理子

●広告
松井竜馬
大橋　涼
北川香織

●発行所
㈱技術評論社
編集部
TEL：03-3513-6170
販売促進部
TEL：03-3513-6150
法人営業課（広告）
TEL：03-3513-6165

●印刷
図書印刷㈱

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。